NXP Semiconductors

Document Number: MPC5777MRM
Rev. 4.3, 01/2017

MPC5777M Reference Manual

This is the MPC5777M Reference Manual set, consisting of the following files:

- MPC5777M Reference Manual, Rev. 4.3
- MPC5777M Reference Manual, Rev. 4.2
- MPC5777M Reference Manual, Rev.4. 1
- MPC5777M Reference Manual, Rev. 4

.

MPC5777M Reference Manual

MPC5777M Reference Manual, Revision 4.3

Document Number: MPC5777MRM
Rev. 4.3, 01/2017

Contents

Section number **Title** **Page**

Chapter 1
Addenda Overview

Chapter 2
RM Addendum Rev 4.3

Chapter 3
RM Addendum Rev 4.2

Chapter 4
RM Addendum Rev 4.1

Chapter 1
Addenda Overview

1.1 Addenda Overview

The table below displays the updates for MPC5777M RM Rev 4.3, RM Rev 4.2 and RM Rev 4.1.

Table 1-1. MPC5777M RM Addenda

Location	Description
MPC5777M RM 4.3 (01/2017)	
	• Updated Reference Manual footer to remove Preliminary.
MPC5777M RM 4.2 (08/2016)	
Chapter 2, Introduction	
Page 124	• Updated the Block Diagram. See Block Diagram
Chapter 5, Memory Map	
Page 153	• Updated the "Used size" of Flash to 512 KB in the table 'Overview memory map'. See Data Flash Blocks–no overlay
Page 167	• Updated the row "0x00400104" in the table 'UTest flash memory map'. See System RAM memory maps
Chapter 7, Device Configuration	
Page 212	• Added 'IPS Clock Gating Module Enable 1 (IPS_CGM_EN1) Register'. See Clock Gating Module Enable 1 (IPS_CGM_EN1)
Page 222	• Changed the address offset of IPS Clock Gating Module Enable 0. See IPS Clock Gating Module Enable 0 (IPS_CGM_EN0)
Page 224	• Added a note in the section 'INTC implemented registers'. See INTC implemented registers
Page 224	• Removed interrupts in 'Table 7-23 Interrupt sources', which are not supported by the device.

Table continues on the next page...

Table 1-1. MPC5777M RM Addenda (continued)

Location	Description
	See Interrupt sources
Page 268	• Corrected the Reset value of SWT_CR0, SWT_CR1, SWT_CR2, and SWT_CR3 registers in the section 'Default configuration'. See Default configuration
Page 273	• Updated the FEC interface selection section and table FEC interface selection. See FEC interface selection
Page 284	• Changed the Auto configuration for linflex_1 to linflex_16 from 'no' to 'n/a'. See LINFlexD configurations
Page 285	• Updated the MREV and JPIN value in the table 'SSCM_MEMCONFIG Reset Values'. See SSCM_MEMCONFIG reset values
Page 287	• Added a new section 'Reset values on BAF exit'. See Reset values on BAF exit
Page 294	• Updated some entries in table 'LVD /HVD self test decoding' column 'LVD / HVD under test'. See LVD /HVD self test decoding
Page 308	• Updated the table 'Availability and reset values of selected FCCU registers'. See FCCU chip-specific register reset values
Page 310	• Updated the figure 'Error signal flow'. See Error signal flow
Page 311	• Updated the footnotes of 'Table 7-86. FCCU failure inputs'. See FCCU failure inputs
Chapter 12, Calibration and Debug	
Page 443	• Added sections DCI signals and DCI EVTx Pin Multiplexing Control Register (DCI_PINCR). See, DCI signals and DCI $\overline{\text{EVTx}}$ Pin Multiplexing Control Register (DCI_PINCR)
Chapter 14, Core description	
Page 532	• Updated the Reset value of DMEM Control Register 0. See DMEM Control register 0
Page 539	• Updated the Reset value of IMEM Control Register 0. See IMEM Control register 0
Chapter 15, Core description	
Page 615	• Updated the Reset value for DMEM Control Register 0.

Table continues on the next page...

Table 1-1. MPC5777M RM Addenda (continued)

Location	Description
	See DMEM Control register 0
Page 623	• Updated the Reset value of IMEM Control Register 0. See IMEM Control register 0
Chapter 17, Crossbar Switch	
Page 771	• Updated the reset value of XBAR Control Register. See, XBAR Control register.
Chapter 23, Interrupt Controller (INTC)	
Page 926	• Updated the section 'Examining LIFO contents'. See, Examining LIFO contents.
Chapter 24, Enhanced Direct Memory Access (eDMA)	
Page 930	• Updated the description of Memory map/register definition section. See Memory map/register definition
Chapter 26, Clocking	
Page 1086	• Updated the table 'Maximum system level clock frequencies'. See System clock frequency limitations
Page 1088	• Updated the table 'JTAG frequencies'. See JTAG frequencies
Page 1088	• Added a new section 'System clock ratio restrictions'. See System Clock ratio restrictions
Page 1097	• Updated the Clock distribution figure. See Clock distribution
Page 1102	• Added a note in the M_TTCAN/M_CAN clocking section. See, M_TTCAN/M_CAN clocking
Page 1108	• Added guideline for PCS register configuration in the 'Progressive clock switching' section. See, Progressive clock switching
Chapter 27, Dual PLL Digital Interface (PLLDIG)	
Page 1118 and Page 1123	• Added a reference to the section 'Clock configuration' in register description of PLLDIG PLL0 Divider Register and PLLDIG PLL1 Divider Register.
Page 1129	• Added equation for f_{pll1_VCO} and f_{pll1_phi} when PLLDIG_PLL1FD[FDEN] = 1b. • Updated equations f_{PLL1_PHI} and f_{pll1_VCO}. • See Clock configuration
Chapter 29, Clocking Generation Module	
Page 1204	• Updated Equation 11.

Table continues on the next page...

Table 1-1. MPC5777M RM Addenda (continued)

Location	Description
	See Generic clock change properties
Chapter 31, IRCOSC Digital Interface	
Page 1230	• Updated the Reset value of 'IRCOSC Control Register (IRCOSC_CTL)'. See IRCOSC Control register
Chapter 34, Embedded Flash Memory (c55fmc)	
Page 1303	• Changed the reset value of the Alternate Module Configuration Register, the footnote in the register diagram replaced with footnotes for Over-Program Protection 0 register, and the Footnote in the register diagram replaced with footnote in Over-Program Protection 1 register, Over-Program Protection 2 register, and Over-Program Protection 3 register. See C55FMC Memory map and register definition
Page 1346	• Updated the existing Note in the section 'Program suspend/resume'. See Program suspend/resume
Page 1352	• Paragraph added to section 'Array integrity self check'. See Array integrity self check
Chapter 37, External Bus Interface	
Page 1418 and 1458	• Removed the column 'EBI_MCR [SIZE]' from the tables 'Table 37-3. Write/ Byte Enable Signals Function', ' Table 37-9. Data Bus Requirements for Read Cycles' and 'Table 37-10. Data Bus Contents for Write Cycles'. See, External Bus Interface
Chapter 38, Analog-to-Digital Converters (ADC) Configuration	
Page 1496	• Updated the figure SAR ADC integration diagram. See SAR ADC integration diagram
Page 1500	• SARADC_B sampling time unit changed. See Self Test features
Page 1547, 1550, and 1553	• Updated the field name 'SARADCx_ICWSELR7' in the tables 'Table 38-31 SARADC_8 register definitions, Table 38-32 SARADC_9 register definitions and Table 38-33 SARADC_10 register definitions'. See SARADC register description
Chapter 39 Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface,	
Page 1572	• Added FRST and FOWEN fields in FIFO Control Register. See FIFO Control register
Chapter 40, Successive Approximation Register Analog-to- Digital Converter (SARADC) Digital Interface	
Page 1651	• Added a Note in the Test Channel Data Register (SARADC_TCDRn) description. See Test Channel Data register
Chapter 48, CAN Subsytem	
Page 1837	• CAN FD support is removed throughout the document.

Table continues on the next page...

Table 1-1. MPC5777M RM Addenda (continued)

Location	Description
	See, CAN Subsystem
Page 1879	• The FOS field width updated to 7 bits for the Rx FIFO 0 Configuration Register . See M_CAN_RXF0C[F0S]
Page 1867	• Added bit field values for the DRXE field in Interrupt Enable Register. See DRXE
Chapter 52, LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications	
Page 2282	• Changed the instances of digrf to LFAST. See, Digrf changed to LFAST
Page 2312	• Fixed typo error, changed the fields name from CTSMN to RCTSMN, CTSMX to RCTSMX, and TISR[CTSMX] to RFCR[CTSMX] in the section CTS mode support. See CTS mode support
Chapter 59, LINFlexD	
Page 3199 and Page 3219	• Updated the field description of IOPE in the LIN Control Register2 and in the LINS field changed the bit field value description of 0001 Init mode in the LIN Status Register. See LINFlexD
Chapter 63 Power Management Controller digital interface	
Page 3407	• Added a sentence in the Reset Event Select Register description. See Reset Event Select register
Chapter 73, JTAG Master	
Page 3956	• For the field TCKSEL updated the TCK divide value. See Module Configuration register
Chapter 76, LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug	
Page 4004	• Changed the instances of digrf to LFAST. See, Digrf changed to LFAST
Page 4066	• Updated the description of the field LVLPEN. See LVLPEN
Chapter 86, Self-Test Control Unit	
Page 4463	• Updated the section 'Register write-access watchdog timer'. See, Register write-access watchdog timer
	• Added a chip specific topic. See STCU2 • Added a chip specific topic for AUTOLOCK_VALUE with info about DCF_COMPLETION.

Table continues on the next page...

Table 1-1. MPC5777M RM Addenda (continued)

Location	Description
	See AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion
Chapter 85, Fault Collection and Control Unit (FCCU)	
Page 4423 and 4447	• Removed a note from the RFS Configuration Register and NMI Enable Register. See FCCU
MPC5777M RM 4.1 (05/2015)	

Chapter 5, Memory Map, page167	• Table 5-6,In the UTest flash memory map, made the following changes for starting address 0x00400308:
	  • Changed the name in the Description column" to "Reserved".
	  • Changed the existing note.
	  • Added an additional note.
	See, UTest flash memory map

Chapter 2
RM Addendum Rev 4.3

2.1 Overview

The update for MPC5777M RM Rev 4.2 is as follows:

- Updated Reference Manual footer to remove Preliminary.

Chapter 3
RM Addendum Rev 4.2

3.1 Overview

The updates for MPC5777M RM Rev 4.1 are described in detail below.

3.1.1 Block Diagram

- The Block Diagram is updated. The 'Concentrator with E2E ECC 50 MHz' changed to 'Concentrator with E2E ECC' and the connection arrows from the FlexRay Ethernet and LFAST & SIPI now have the frequency listed on them.

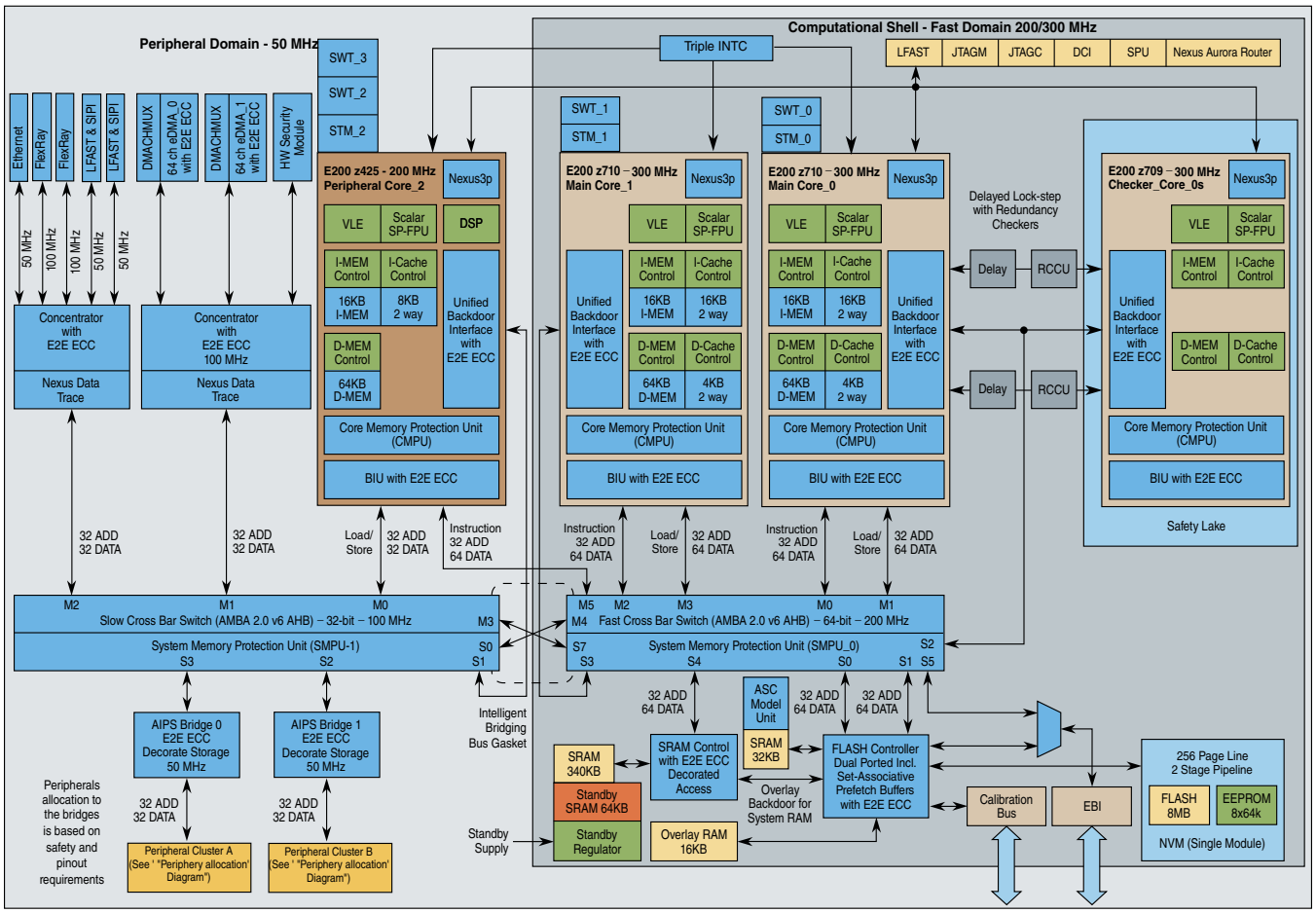


Figure 3-1. Block diagram

3.1.2 Memory Map

3.1.2.1 Data Flash Blocks–no overlay

- In Table 3-1, for the starting address 0x00800000, changed the "Used size" to 512 KB.

Table 3-1. Overview memory map

Start address	End address	Allocated size	Used size	Description
Flash (XBAR port 0—0)				
0x00800000	0x009FFFFF	2 MB	512 KB	Data Flash Blocks–no overlay

3.1.2.2 System RAM memory maps

- Updated the 0x00400104 row in table UTest flash memory map:

Table 3-2. UTest flash memory map

Start address	End address	Allocated size [bytes]	Description	Comments
0x00400104	0x0040011F	28	Fuse Bypass Password (FA flash test password)	The FA flash test password must be left erased unless the application requires an additional password for testing the flash during failure analysis. If a password is required, additional programming is necessary. See the Security Reference Manual entry for "Fuse Bypass Password" for more information.

3.1.3 Device configuration

3.1.3.1 Clock Gating Module Enable 1 (IPS_CGM_EN1)

- Clock Gating Module Enable 0 (IPS_CGM_EN0) register is added under the section Platform Configuration Module.

Offset 18h Access: Supervisor read/write

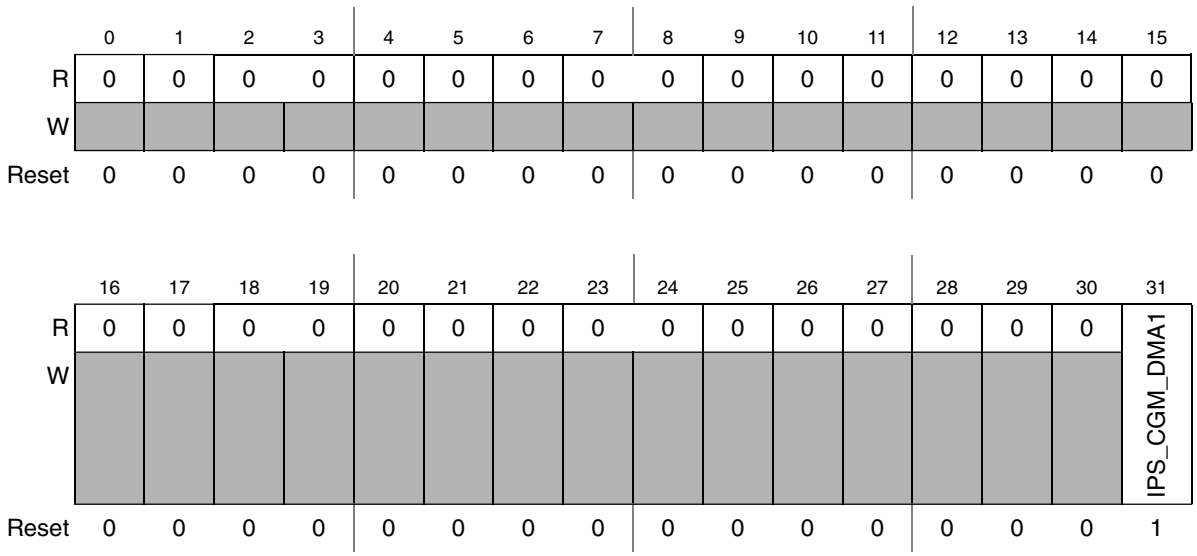


Figure 3-2. IPS Clock Gating Enable 1 (IPS_CGM_EN1) register

Table 3-3. IPS_CGM_EN1 field description

Field	Description
0–30	Reserved
31 IPS_CGM_DMA1	IPS clock gating enable for DMA1 1 Enable clock gating, one IPS clock delay is introduced between master and slave. 0 Disable clock gating.

3.1.3.2 IPS Clock Gating Module Enable 0 (IPS_CGM_EN0)

- Changed the address offset of IPS Clock Gating Module Enable 0 register to 0x14h.

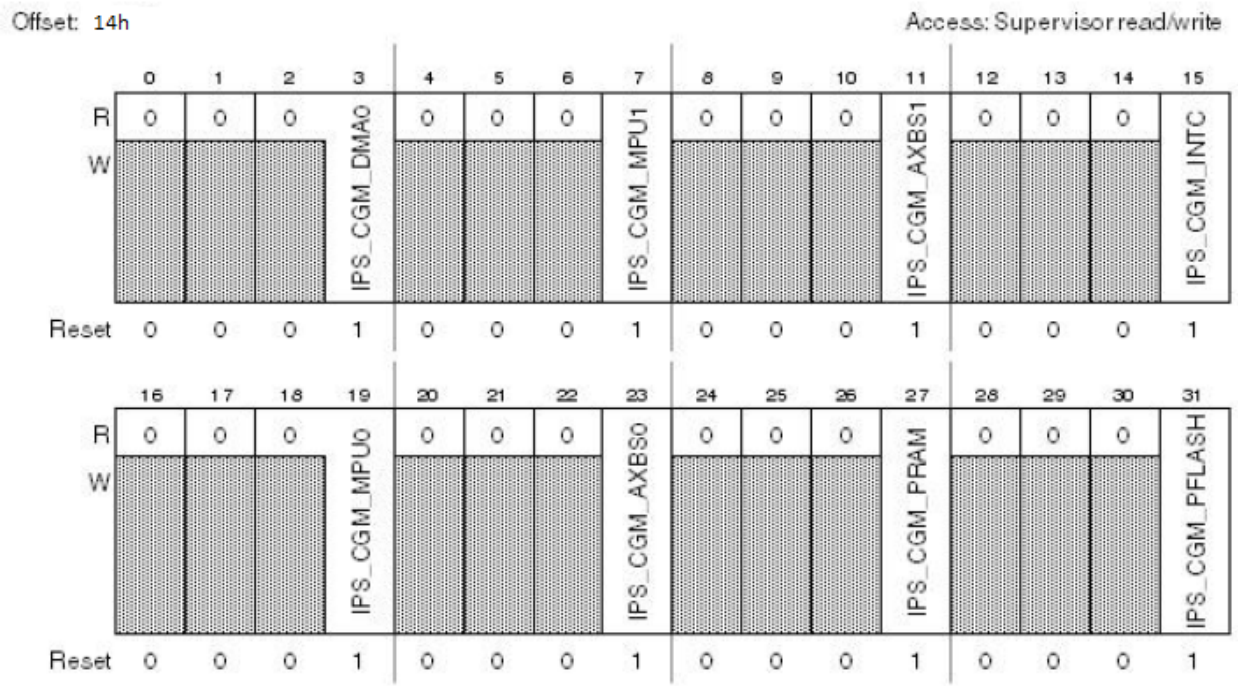


Figure 3-3. IPS Clock Gating Module Enable 0 register

3.1.3.3 INTC implemented registers

- The following note has been added to the section:

NOTE

There are four cores implemented on this chip, but core 3 is strictly a checker core. The INTC only supports the three functional cores 0, 1, and 2. Therefore, the registers INTC_IACKR3, INTC_CPR3, and INTC_EOIR3 are not supported, and any attempt to access these registers results in a bus error. Also, writing to the HVEN3 bit in the INTC_BCR register has no effect.

3.1.3.4 Interrupt sources

- Removed the following interrupts not supported by the device from 'Table 7-23 Interrupt sources':
  - DSPI_0_SR(TFIWF)
  - DSPI_1_SR(TFIWF)

- DSPI_2_SR(TFIWF)
- DSPI_3_SR(TFIWF)
- DSPI_4_SR(TFIWF)
- DSPI_5_SR(TFIWF)
- DSPI_6_SR(TFIWF)
- DSPI_12_SR(TFIWF)

3.1.3.5 Default configuration

- Changed the reset values of SWT_CR [0, 1, 2, 3] registers.
  For SWT[0, 1, 3], the SWT_CR register reset value of 0xFF00_010A selects:
  - All masters allowed access
  - Reset on invalid access
  - Oscillator clock selected
  - Counter stops in debug mode
  - Watchdog is disabled

  For SWT[2], the SWT_CR register reset value of 0xFF00_011B selects:
  - All masters allowed access
  - Reset on invalid access
  - Oscillator clock selected
  - Counter stops in debug mode
  - Soft locked
  - Watchdog is enabled

3.1.3.6 FEC interface selection

- On this chip, the FEC interface depends on the settings of FEC_RCR[MII_MODE] (in the FEC) and the SoC Configuration Register0 SIUL2_SCR0[FEC_MODE].

Table 3-4. FEC interface selection

FEC_RCR[MII_MODE]	SIUL2_SCR0[FEC_MODE]	Interface used by the FEC
0	0 or 1	7-wire
1	0	RMII
	1	MII

3.1.3.7 LINFlexD configurations

- The Auto synchronization for Linflex1-16 changed from 'no' to 'n/a' in LINFlexD configurations table.

Table 3-5. LINFlexD configurations

Description	linflex_0	linflex_1	linflex_2	linflex_14	linflex_15	linflex_16
Number of filters implemented	16	0	0	0	0	0
Number of Tx DMA channels	1	1	1	1	1	1
Number of Rx DMA channels	1l	1	1	1	1	1
LIN operation mode	master/ slave	master	master	master	master	master
Auto synchronization	yes	n/a	n/a	n/a	n/a	n/a

3.1.3.8 SSCM_MEMCONFIG reset values

- In the table SSCM_MEMCONFIG reset values' updated the MREV and JPIN values.

Table 3-6. SSCM_MEMCONFIG reset values

Field	Field Definition	Value
MREV	Minor Revision	0001b
JPIN	JTAG Part ID	30Fh

3.1.3.9 Reset values on BAF exit

Added the section reset values on BAF exit under the section Boot Assist Flash (BAF) configuration.

- BAF interacts with various modules during its course of execution. BAF enables and disables the reset of values of registers for these modules. The reset values of some status registers which are not restored by BAF on exit are listed in the table below.

Table 3-7. Reset Values at Serial Boot

Sl. No.	Module Name	Register	BAF Exit Reset Value
1	MC_ME	MC_ME_RUN_PC_0	0xFE
2	MC_ME	MC_ME_DRUN	0x130072
3	MC_ME	MC_ME_GS	0x130072
4	MC_CGM	MC_CGM_SC_DC2	0x80010000
5	MC_CGM	MC_CGM_AC3_SC	0x0

Table continues on the next page...

Table 3-7. Reset Values at Serial Boot (continued)

Sl. No.	Module Name	Register	BAF Exit Reset Value
6	MC_CGM	MC_CGM_AC0_SC	0x02000000
7	MC_CGM	MC_CGM_AC0_DC4	0x80000000
8	MC_CGM	MC_CGM_AC8_SC	0x01000000
9	MC_CGM	MC_CGM_AC8_DC0	0x80000000
10	PLLDIG	PLLDIG_PLL0DV	0x00061021
11	FCCU	FCCU_RF_CFG_0	0xFFFFEFFF

Table 3-8. Reset Values at Flash Boot

Sl. No.	Module Name	Register	BAF Exit Reset Value
1	MC_ME	ME_IS	0x00000001
2	MC_ME	ME_DMTS	0x30000000
3	FCCU	FCCU_RF_CFG_0	0xFFFFEFFF

3.1.3.10 LVD /HVD self test decoding

- Entries updated under column 'LVD / HVD under test' in the table LVD /HVD self test decoding:

Table 3-9. LVD /HVD self test decoding

Current Name	New Name
POR098_C	LVD096_C
LVD114_C	LVD112_C
POR098_F	LVD096_F
LVD114_F	LVD108_F
LVD114_P	LVD108_P
LVD114_H	LVD108_C
LVD280_IF2	LVD270_IF2
LVD280_IE	LVD270_IE
LVD280_C	LVD270_C
LVD280_IF	LVD270_IF
LVD280_IM	LVD270_IM
LVD280_IJ	LVD270_IJ
LVD280_O	LVD270_O
LVD300_F	LVD295_F
LVD300_A	LVD295_A

3.1.3.11 FCCU chip-specific register reset values

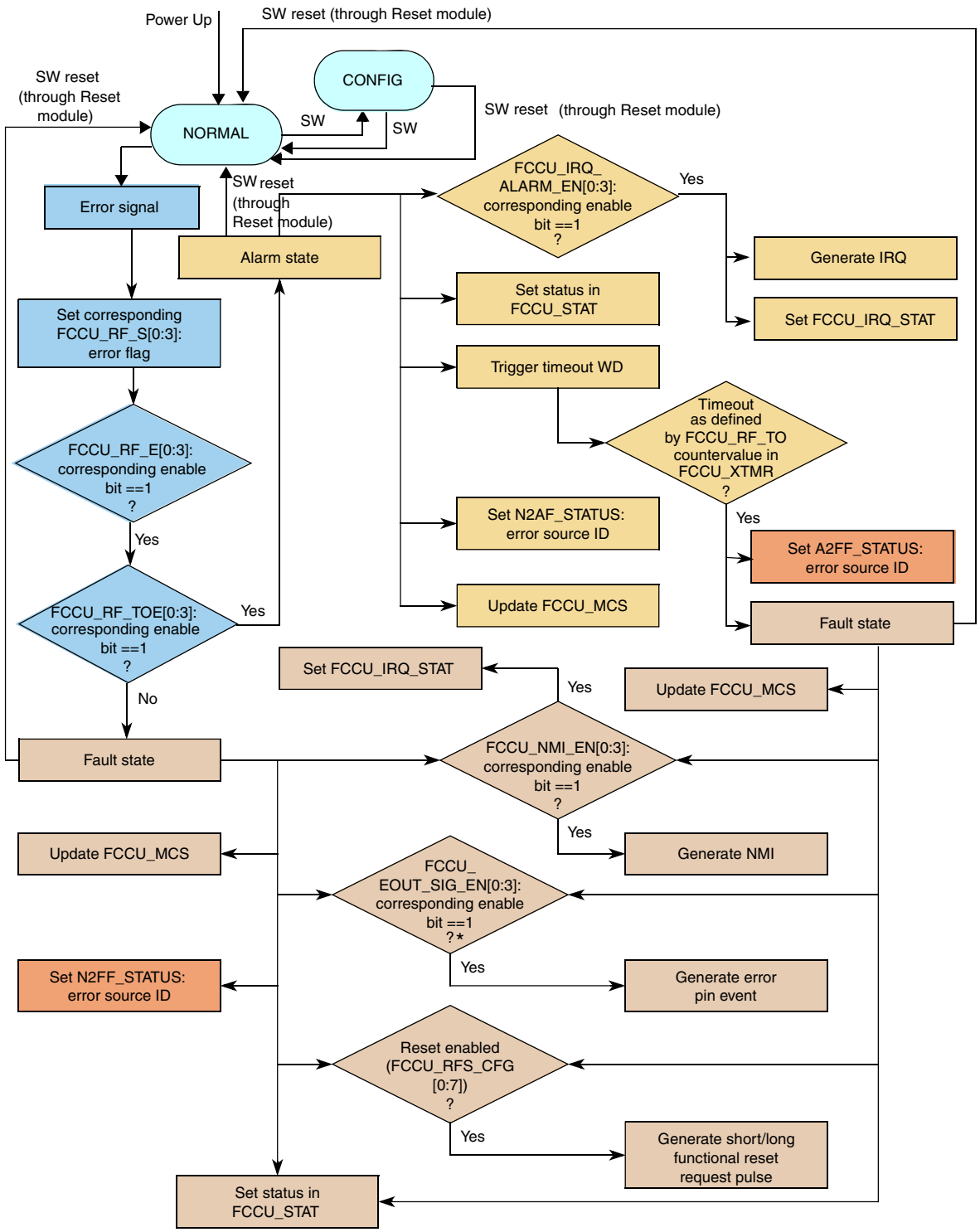
This table lists in alphabetical order by mnemonic the FCCU registers whose reset values are specific to this chip.

Table 3-10. Availability and reset values of selected FCCU registers

Register	Reset value
FCCU_CFG	0x0000_0000
FCCU_CFG_TO	0x0000_0006
FCCU_MCS	0x0000_0000
FCCU_RF_CFG0	0xFFFF_FFFF
FCCU_RF_CFG1	0xFFFF_FFFF
FCCU_RF_CFG2	0xFFFF_FFFF
FCCU_RF_CFG3	0xFFFF_FFFF
FCCU_RF_E0	0xFFFF_FFFF
FCCU_RF_E1	0xFFFF_FFFF
FCCU_RF_E2	0xFFFF_FFFF
FCCU_RF_E3	0xFFFF_FFFF
FCCU_RF_TO	0x0000_FFFF
FCCU_RF_TOE0	0xFFFF_FFFF
FCCU_RF_TOE1	0xFFFF_FFFF
FCCU_RF_TOE2	0xFFFF_FFFF
FCCU_RF_TOE3	0xFFFF_FFFF
FCCU_RFS_CFG0	0x0000_0000
FCCU_RFS_CFG1	0x0000_0000
FCCU_RFS_CFG2	0x0000_0000
FCCU_RFS_CFG3	0x0000_0000
FCCU_RFS_CFG4	0x0000_0000
FCCU_RFS_CFG5	0x0000_0000
FCCU_RFS_CFG6	0x0000_0000
FCCU_RFS_CFG7	0x0000_0000

3.1.3.12 Error signal flow

In the Figure 3-4, added a footnote 'This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM])'.



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 3-4. Error signal flow

3.1.3.13 FCCU failure inputs

- Editorial updates in the footnotes of Table 7-86. FCCU failure inputs:
  - Added cross reference to Figure 26-1 and Figure 26-2 Clock generation in footnote number 12.
  - Removed the footnote number 14.
  - Changed 200z4 Core to z7 core in footnote number 18.
  - For Channel 31 updated the footnotes to Channel 31 [1,13].

3.1.4 DCI signals

The DCI sends and receives the $\overline{\text{EVTI}}$[1:0] and $\overline{\text{EVTO}}$[1:0] signals to/from the cores and other modules. The tables below summarize the source and destination of $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, and related signals for the PD and the BD. For example, for the PD, $\overline{\text{EVTI}}$[0] are outputs of the JTAGM and SPU and inputs to the cores, HSM, GTM, and NXMC.

Table 3-11. PD DCI signals

Signal	Cores	HSM	GTM	NXMC	NAR	JTAGM	SPU	DTS
$\overline{\text{EVTI}}$[0]	I	I	I	I	—	O	I/O	—
$\overline{\text{EVTI}}$[1]	—	—	—	—	—	O	I/O	—
$\overline{\text{EVTO}}$[0]	O	O	O	O	—	I[1]	I/O	—
$\overline{\text{EVTO}}$[1]	—	—	—	—	—	I[1]	I/O	O
Core debug event	I	—	—	—	—	—	—	—
DCI cross triggering	O	—	O	—	—	—	—	—

1. The EVTO signal provided to the JTAGM is stretched by the DCI on the 16MHz IRC clock. Therefore, the incoming EVTO cannot be reliably detected by the JTAGM when the JTAGM clock is configured at less than 32MHz.

Table 3-12. BD DCI signals

Signal	RWA	NAR	NAL	JTAGM
$\overline{\text{EVTI}}$[0]	—	I	—	O
$\overline{\text{EVTI}}$[1]	—	—	—	O
$\overline{\text{EVTO}}$[0]	—	—	—	I[1]
$\overline{\text{EVTO}}$[1]	—	—	—	I[1]

1. The EVTO signal provided to the JTAGM is stretched by the DCI on the 16MHz IRC clock. Therefore, the incoming EVTO cannot be reliably detected by the JTAGM when the JTAGM clock is configured at less than 32MHz.

The DCI sends the system debug signal, ipg_debug, to the modules below. For details regarding module functionality during system debug, refer to the individual module chapter.

- AMU
- CAN
- LFAST (Interprocessor Communications instance only)
- DSPI
- eDMA
- FCCU
- GTMINT
- I2C
- NXMC
- PIT
- PSI5
- SARADC
- SDADC
- SRX
- SSCM
- STM
- SWT

3.1.4.1 DCI $\overline{\text{EVTx}}$ Pin Multiplexing Control Register (DCI_PINCR)

The DCI $\overline{\text{EVTx}}$ Pin Multiplexing Control Register (DCI_PINCR) is described below. The $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ functions share the same set of possible pins. If this register is written to enable both $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ on the same pin, the $\overline{\text{EVTI}}$ function is selected, although this type of programming should be avoided. Programming the $\overline{\text{EVTI}}$ fields to select $\overline{\text{EVTI}}$[0] or $\overline{\text{EVTI}}$[1] inputs to come from more than one pin is considered a programming error and no pin is enabled.

	31	30	29	28	27	26	25	24
R W	Reserved							EVTO0E

Table continues on the next page...

Reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	Reserved			EVTI1E	Reserved			EVTI0E
Reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	EVTO1D	EVTO1C	EVTO1B	EVTO1A	EVTO0D	EVTO0C	EVTO0B	EVTO0A
Reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	EVTI1D	EVTI1C	EVTI1B	EVTI1A	EVTI0D	EVTI0C	EVTI0B	EVTI0A
Reset	0	0	0	0	0	0	0	0

Table 3-13. DCI_PINCR field descriptions

Field	Description
24 EVTO0E	Enables $\overline{\text{EVTO}}$[0] function on pin PA[8]
20 EVTI1E	Enables $\overline{\text{EVTI}}$[1] function on pin PA[9]
16 EVTI0E	Enables $\overline{\text{EVTI}}$[0] function on pin PA[8]
15 EVTO1D	Enables $\overline{\text{EVTO}}$[1] function on pin PA[9]
14 EVTO1C	Enables $\overline{\text{EVTO}}$[1] function on pin PM[6]
13 EVTO1B	Enables $\overline{\text{EVTO}}$[1] function on pin PK[14]
12 EVTO1A	Enables $\overline{\text{EVTO}}$[1] function on pin PH[11]
11 EVTO0D	Enables $\overline{\text{EVTO}}$[0] function on pin PM[5]
10 EVTO0C	Enables $\overline{\text{EVTO}}$[0] function on pin PM[4]
9 EVTO0B	Enables $\overline{\text{EVTO}}$[0] function on pin PF[15]
8	Enables $\overline{\text{EVTO}}$[0] function on pin PF[14]

Table continues on the next page...

Table 3-13. DCI_PINCR field descriptions (continued)

Field	Description
EVTO0A	
7 EVTI1D	Enables $\overline{\text{EVTI}}$[1] function on pin PM[7]
6 EVTI1C	Enables $\overline{\text{EVTI}}$[1] function on pin PM[6]
5 EVTI1B	Enables $\overline{\text{EVTI}}$[1] function on pin PK[14]
4 EVTI1A	Enables $\overline{\text{EVTI}}$[1] function on pin PH[11]
3 EVTI0D	Enables $\overline{\text{EVTI}}$[0] function on pin PM[5]
2 EVTI0C	Enables $\overline{\text{EVTI}}$[0] function on pin PM[4]
1 EVTI0B	Enables $\overline{\text{EVTI}}$[0] function on pin PF[15]
0 EVTI0A	Enables $\overline{\text{EVTI}}$[0] function on pin PF[14]

3.1.5 Core description

3.1.5.1 DMEM Control register 0

- The Reset value for DMEM Control Register 0 is updated to 0x0000_041C.

DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

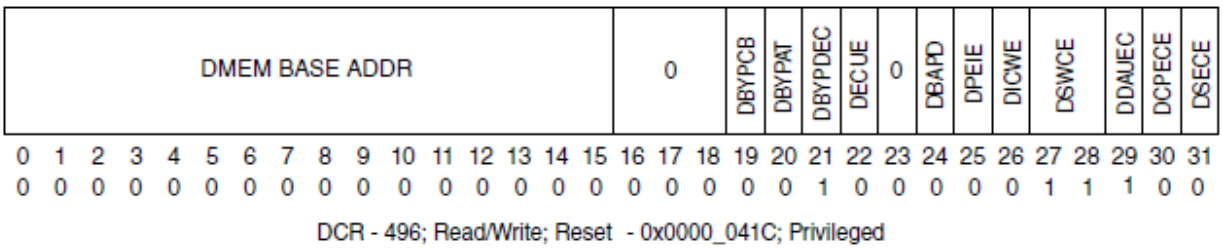


Figure 3-5. DMEM Control register 0 (DMEMCTL0)

- Updated the Note to 'This field is updated by an internal Power_On_Reset signal or an Internal Destructive Reset.' in the field description of DBYPCB, DBYPAT, DBYPDEC, DECUE, DSWCE, DDAUEC, DCPECE, and DSECE fields.

3.1.5.2 IMEM Control register 0

- The reset value of IMEM Control register 0 updated to 0x0000_0000.

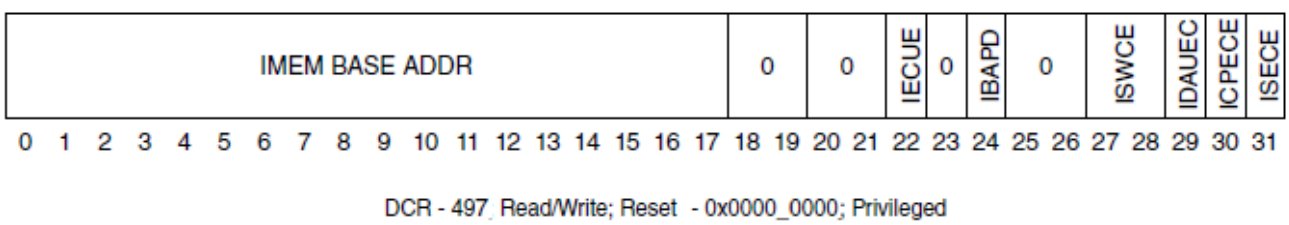


Figure 3-6. IMEM Control register 0

NOTE

This register is reset by an internal Power-On-Reset signal or by an internal destructive reset signal. It is unaffected by p_reset_b.

- Updated the field description of IECUE field.
  - This field is updated by an internal Power_On_Reset signal or an internal destructive reset.

- Updated the Note in the field description of ISWCE, IDAUEC, ICPECE, and ISECE fields.

- This field is updated by an internal Power_On_Reset signal or an internal destructive reset.

3.1.6 Core description

3.1.6.1 DMEM Control register 0

- The Reset value for DMEM Control Register 0 is updated to 0x0000_041C.

DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

DMEM BASE ADDR	0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCPECE	DSECE
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	0	0	1	0	0	0	0	1 1	1	0	0

DCR - 496; Read/Write; Reset - 0x0000_041C; Privileged

Figure 3-7. DMEM Control register 0 (DMEMCTL0)

- Updated the Note in the field description of fields DBYPCB, DBYPAT, DBYPDEC, DECUE, DSWCE, DDAUEC, DCPECE, and DSECE.
  - This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset.

3.1.6.2 IMEM Control register 0

- The reset value of IMEM Control register 0 updated to 0x0000_001B.

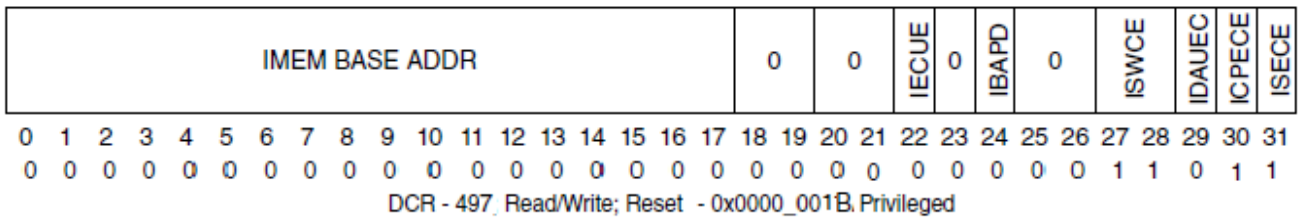


Figure 3-8. IMEM Control register 0

NOTE

This register is reset by an internal Power-On-Reset signal or by an internal destructive reset signal. It is unaffected by p_reset_b.

- Updated the field description of the IECUE field.
  - This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset.

- Updated the 'Note' in the field description of ISWCE, IDAUEC, ICPECE, and ISECE fields.
  - This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset.

3.1.7 XBAR Control register

- Changed the reset value of the XBAR Control register to 0x003F_0000 (previously it was 0x00FF_0000).

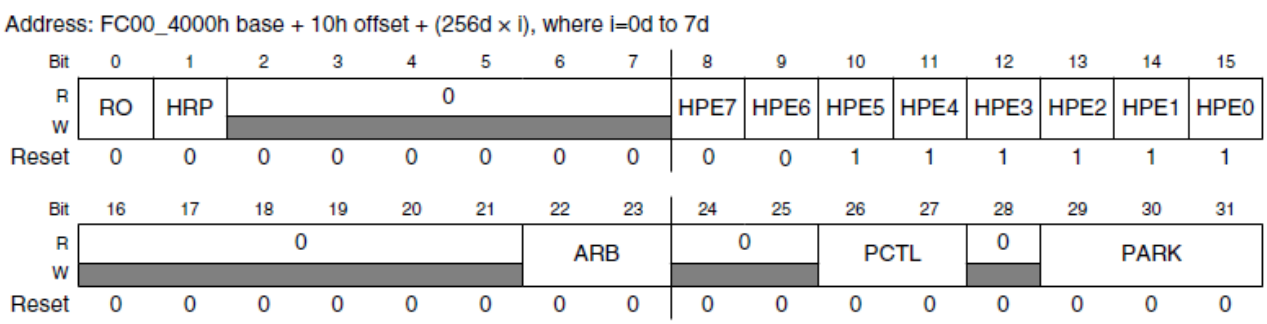


Figure 3-9. XBAR Control register

3.1.8 Examining LIFO contents

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRIN field in the INTC current priority register (INTC_CPRn). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPRn, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
     wrteei 0                # disable processor recognition of external
interrupts
     oldCPRn = INTC_CPRn;    # save INTC_CPRn
      (branch to the pop_lifo)
pop_lifo:
     store INTC_EIORn        # pop INTC_CPR from LIFO, examine PRI N, etc...
     examine INTC_CPRn[PRI N], and store onto stack
     if PRI N is not zero or value when interrupts were enabled, branch to pop_lifo
     branch to push_lifo
```

A side effect of examining LIFO contents is the clearing of a pending SWTN field without triggering the interrupt. To compensate for this issue, you must load the PSR again to restore the SWTN field.

The following describes the restoring of the LIFO and the SWTN field:

```
push_lifo:

load stacked INTC_PSRn[PRIN] value and store to INTC_CPRn

read INTC_IACKRn # Pushes INTC_CPRn to LIFO but also clears INTC_PSRn[SWTN]

load INTC_PSRn again to restore INTC_PSRn[SWTN]

if stacked INTC_PSRn[PRI] values are not depleted, branch to push_lifo
```

NOTE

Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

3.1.9 Memory map/register definition

In the memory map/register description added information about TCD.

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions.
- The second region corresponds to the local transfer control descriptor memory.

TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCDn definition is presented as 11 registers of 16 or 32 bits.

TCD initialization

The initialization of the TCD memory is performed by the eDMA controller after reset. The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

TCD structure

DMA Basics: TCD Structure
- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD

Word Offset	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
0x1000	SADDR							
0x1004	SMOD	SSIZE	DMOD	DSIZE	SOFF			
0x1008	NBYTES[1]							
0x1008	SMLOE[1] / DMLOE[1]	MLOFF or NBYTES[1]				NBYTES[1]		
0x100C	SLAST							
0x1010	DADDR							
0x1014	CITER.E_LINK	CITER or CITER.LINKCH	CITER		DOFF			
0x1018	DLAST_SGA							
0x101C	BITER.E_LINK	BITER or BITER.LINKCH	BITER		BWC	MAJOR LINKCH	DONE / ACTIVE / MAJOR.E_LINK / E_SG / D_REQ / INT_HALF / INT_MAJ / START	
	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31

[1] The fields implemented in Word 2 depend on whether EDMA_CR[EMLM] bit is set to 0 or 1.

3.1.10 Clocking

3.1.10.1 System clock frequency limitations

- Updated the SAR_CLK to 14.6 MHz

Table 3-14. Maximum system level clock frequencies

Clock Selector	Nominal programmed Max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 0 Selector	400	0	PER_CLK	80
		1	SD_CLK	16
		2	SAR_CLK	14.6

3.1.10.2 JTAG frequencies

- Updated the e200z7 max. internal JTAG frequency to 100 MHz for PD.

Table 3-15. JTAG frequencies

Device	Configuration	e200z7 max. internal JTAG frequency	Buddy device max. RWA internal frequency	Max. external JTAG frequency	Notes
PD	Unconfigured clock	—	N/A	See electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock on PD (IRC) with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
	Clock configured to 300 MHz	100 MHz			

3.1.10.3 System Clock ratio restrictions

The device supports the following clock ratios:

- COMP_CLK/CHKR_CLK : FXBAR_CLK - 1:1, 3:2
- FXBAR_CLK : SXBAR_CLK - 1:1, 2:1
- SXBAR_CLK : PBRIDGEA_CLK/PBRIDGEB_CLK - 1:1, 2:1
- SXBAR_CLK : HSM_CLK - 1:1, 2:1, 4:1
- FXBAR_CLK : CLKOUT - 1:1, 2:1, 3:1, 4:1, 5:1, 6:1, 7:1, 8:1

NOTE

When changing system clock ratios, the Divider Update Trigger (MC_CGM_DIV_UPD_TRIG) and System Clock Divider Ratio Change (SYS_DIV_RATIO_CHNG) must be used. This assures that the system is properly halted by hardware to allow the clocks to be correctly updated.

3.1.10.4 Clock distribution

- The figure Clock Distribution is updated. Changed the BIU Module Clock FlexRay (2) Protocol Clock connection from PBRIDGEx_CLK to SXBAR_CLK.

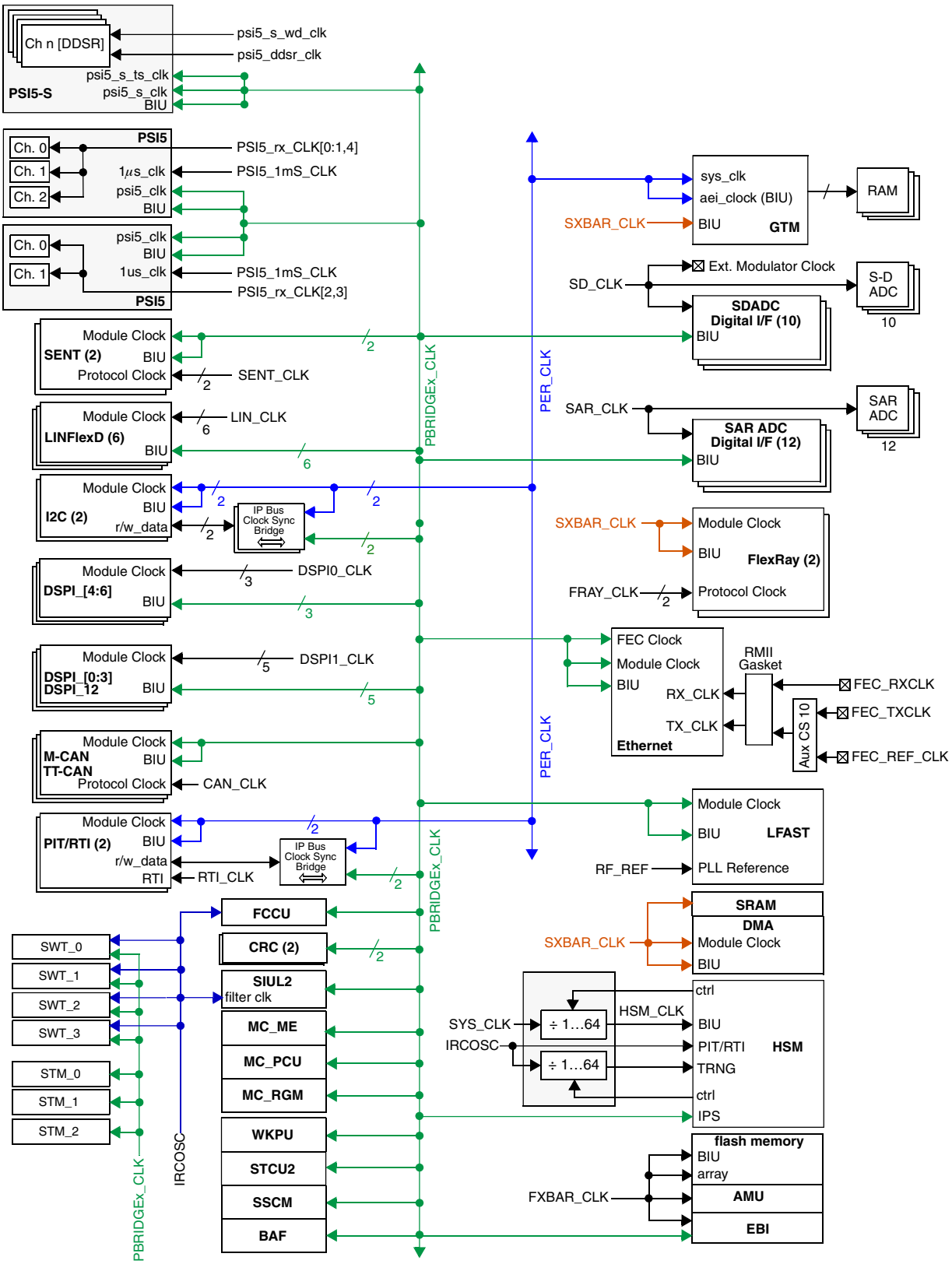


Figure 3-10. Clock distribution

3.1.10.5 M_TTCAN/M_CAN clocking

- Added a note in the section M_TTCAN/M_CAN clocking:
  - The CAN clock frequency needs to be 8 MHz minimum in order to wake up from STOP mode.

3.1.10.6 Progressive clock switching

- Added guideline for PCS register configuration:
  - It is recommended to program the PCS configuration registers (i.e. MC_CGM_PCS_SDUR, MC_CGM_PCS_DIVCn, MC_CGM_PCS_DIVEn and MC_CGM_DIVSn) with IRC selected as the system clock. This insures that during the PCS register programming if a SAFE mode or RESET event happens then there is reliable progressive clock operation.

3.1.11 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV, and PLL1FD registers, and calculated according to the following equations:

$$f_{pll0\_phi} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]} \right)$$

Equation 1. PLL0 PHI Output Frequency

$$f_{pll0\_phi1} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]} \right)$$

Equation 2. PLL0 PHI1 Output Frequency

$$f_{pll1\_phi} = f_{pll1\_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 3. PLL1 PHI Output Frequency (PLLDIG_PLL1FD[FDEN] = 1b)

$$f_{pll1\_phi} = f_{pll1\_ref} \times \left(\frac{PLL1DV[MFD]}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 4. PLL1 PHI Output Frequency (PLLDIG_PLL1FD[FDEN] = 0b)

The relationship between the VCO frequency (fVCO) and the output frequency of the PLLs is determined by the configuration of the PLL1DV, PLL1FD, and PLL0DV registers, according to the following equations:

$$f_{pll0\_VCO} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}\right)$$

Equation 5. PLL0 VCO Frequency

$$f_{pll1\_VCO} = f_{pll1\_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}}\right)$$

Equation 6. PLL1 VCO Frequency (PLLDIG_PLL1FD[FDEN] = 1b)

$$f_{pll1\_VCO} = f_{pll1\_ref} \times PLL1DV[MFD]$$

Equation 7. PLL1 VCO Frequency (PLLDIG_PLL1FD[FDEN] = 0b)

NOTE

f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of the analog PLL blocks to go below the prescribed ranges.

The lock signal and PLLnSR[LOCK] flag are immediately negated if the fields of the PLLnDV registers are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLLnSR[LOCK] flag is held negated. After the timer expires, the PLLnSR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLLnCR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

NOTE

The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_mode_MC before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the input clocks are modified.

3.1.12 Generic clock change properties

Updated the equation in the Generic clock change properties section.

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(f_{src}/(16MHz/1000)\right)}{d}} \right\rceil$$

Equation 1. Equation for k

3.1.13 IRCOSC Control register

- Updated the reset value of IRCOSC Control register from 0x0100_0000 to 0x0000_0000.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							Reserved	0			USER_TRIM				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			Reserved					0		Reserved	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-11. IRCOSC Control register

3.1.14 Embedded Flash Memory (c55fmc)

3.1.14.1 Program suspend/resume

- Updated the existing note:

NOTE

Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend frequency is not limited, enabling at least 20 µs between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

3.1.14.2 Array integrity self check

- Added a new paragraph to the section - 'Array integrity processing requires the flash memory controller's PFLASH_PFCR1[RWSC] and PFLASH_PFCR1[APC] register fields to be set to values that are valid for the system frequency being used. Only the valid/recommended RWSC and APC values for a given frequency, as stated in the device data sheet, are supported.'

3.1.14.3 C55FMC Memory map and register definition

- Reset value of the Alternate Module Configuration Register was changed to 0000_0600h, and the footnote was removed from the register description and diagram.

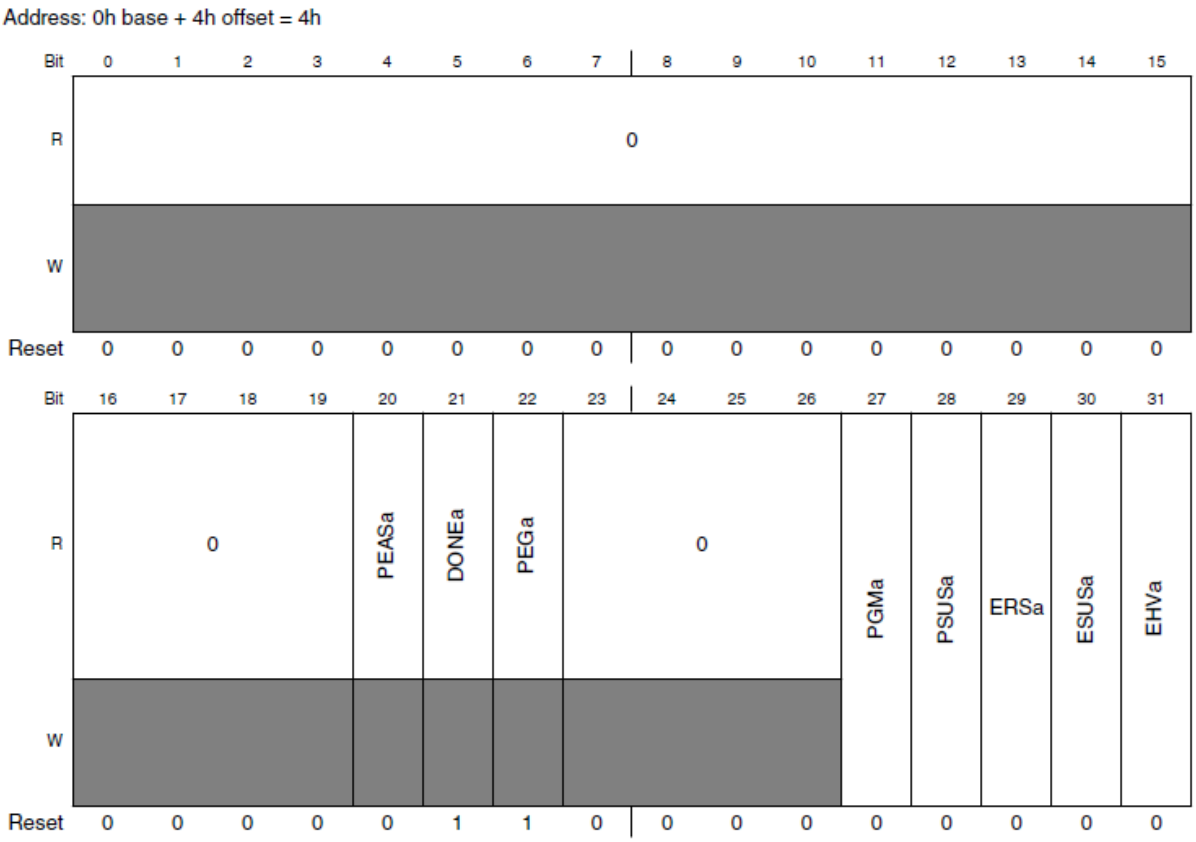
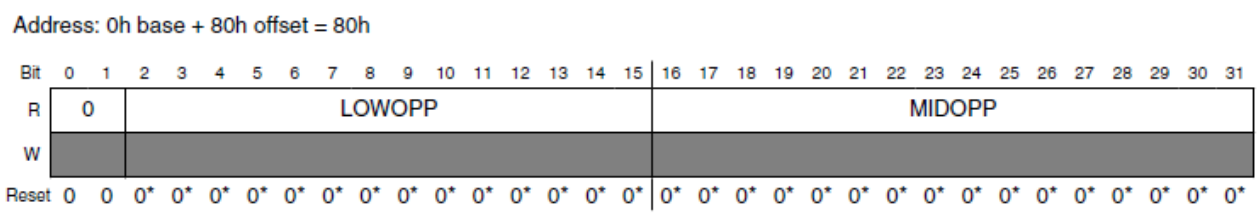


Figure 3-12. Alternate Module Configuration register

- In the Over-Program-Protection 0 register, replaced the footnote in the register diagram with footnotes stating that the reset value of each field (LOWOPP and MIDOPP) varies among devices..



* Notes:

- MIDOPP field: The reset value of the MIDOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.
- LOWOPP field: The reset value of the LOWOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Figure 3-13. Over-Program Protection 0 register

- In the Over-Program-Protection 1 register, replaced the footnote in the register diagram with a footnote stating that the reset value of the HIGHOPP field varies among devices.

Address: 0h base + 84h offset = 84h

Bit	0–15	16–31
R	0	HIGHOPP
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0*

* Notes:
- HIGHOPP field: The reset value of the HIGHOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Figure 3-14. Over-Program Protection 1 register

- In the Over-Program-Protection 2 register, replaced the footnote in the register diagram with a footnote stating that the reset value of the A256OPP field varies among devices.

Address: 0h base + 88h offset = 88h

Bit	0–15	16–31
R	A256KOPP	
W		
Reset	0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0*	0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0*

* Notes:
- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Figure 3-15. Over-Program Protection 2 register

- In the Over-Program-Protection 3 register, replaced the footnote in the register diagram with a footnote stating that the reset value of the A256KOPP field varies among devices.

Address: 0h base + 8Ch offset = 8Ch

Bit	0–15	16–31
R	0	A256KOPP
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0*

* Notes:
- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Figure 3-16. Over-Program Protection 3 register

3.1.15 External Bus Interface

- Removed the references to EBI_MCR[SIZE] from the following tables:

Table 3-16. Write/Byte Enable Signals Function

Transfer Size	Address		32-Bit Port Size				16-Bit Port Size[1]			
	A30	A31	WE0/BE0	WE1/BE1	WE2 /BE2	WE3 /BE3	WE0 /BE0	WE1/BE 1	WE2 /BE2	WE3 /BE3
Byte	0	0	X				X			
	0	1		X				X		
	1	0			X		X			
	1	1				X		X		
16-bit	0	0	X	X			X	X		
	1	0			X	X	X	X		
32-bit	0	0	X	X	X	X	X[-1]	X[-1]		
Burst	0	0	X	X	X	X	X	X		

1. Also applies when DBM=1 for 16-bit data bus mode.

Table 3-17. Data Bus Requirements for Read Cycles

Transfer Size	Address		32-Bit Port Size				16-Bit Port Size[1]	
	A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7[2]	D8:D15[3]
Byte	0	0	OP0	-	-	-	OP0	-
	0	1	-	OP1	-	-	-	OP1
	1	0	-	-	OP2	-	OP2	-
	1	1	-	-	-	OP3	-	OP3
16-bit	0	0	OP0	OP1	-	-	OP0	OP1
	1	0	-	-	OP2	OP3	OP2	OP3
32-bit	0	0	OP0	OP1	OP2	OP3	OP0/OP2[4]	OP1/OP3

1. Also applies when DBM=1 for 16-bit data bus mode.
2. For address/data muxed transfers, DATA[16:23] are used externally, not DATA[0:7].
3. For address/data muxed transfers, DATA[24:31] are used externally, not DATA[8:15].
4. This case consists of two 16-bit external transactions, the first writing OP0 and OP1, and the second writing OP2 and OP3.

Table 3-18. Data Bus Contents for Write Cycles

Transfer Size	Address		32-Bit Port Size				16-Bit Port Size[1]	
	A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7[2]	D8:D15[3]
Byte	0	0	OP0	-	-	-	OP0	-
	0	1	-	OP1	-	-	-	OP1
	1	0	-	-	OP2	-	OP2	-
	1	1	-	-	-	OP3	-	OP3
16-bit	0	0	OP0	OP1	-	-	OP0	OP1
	1	0	-	-	OP2	OP3	OP2	OP3
32-bit	0	0	OP0	OP1	OP2	OP3	OP0/OP2[4]	OP1/OP3

1. Also applies when DBM=1 for 16-bit data bus mode.
2. For address/data muxed transfers, DATA[16:23] are used externally, not DATA[0:7].
3. For address/data muxed transfers, DATA[24:31] are used externally, not DATA[8:15].

4. This case consists of two 16-bit external transactions, the first writing OP0 and OP1, and the second writing OP2 and OP3.

3.1.16 Analog-to-Digital Converters (ADC) configuration

3.1.16.1 SAR ADC integration diagram

- Updated the figure SAR ADC integration diagram.

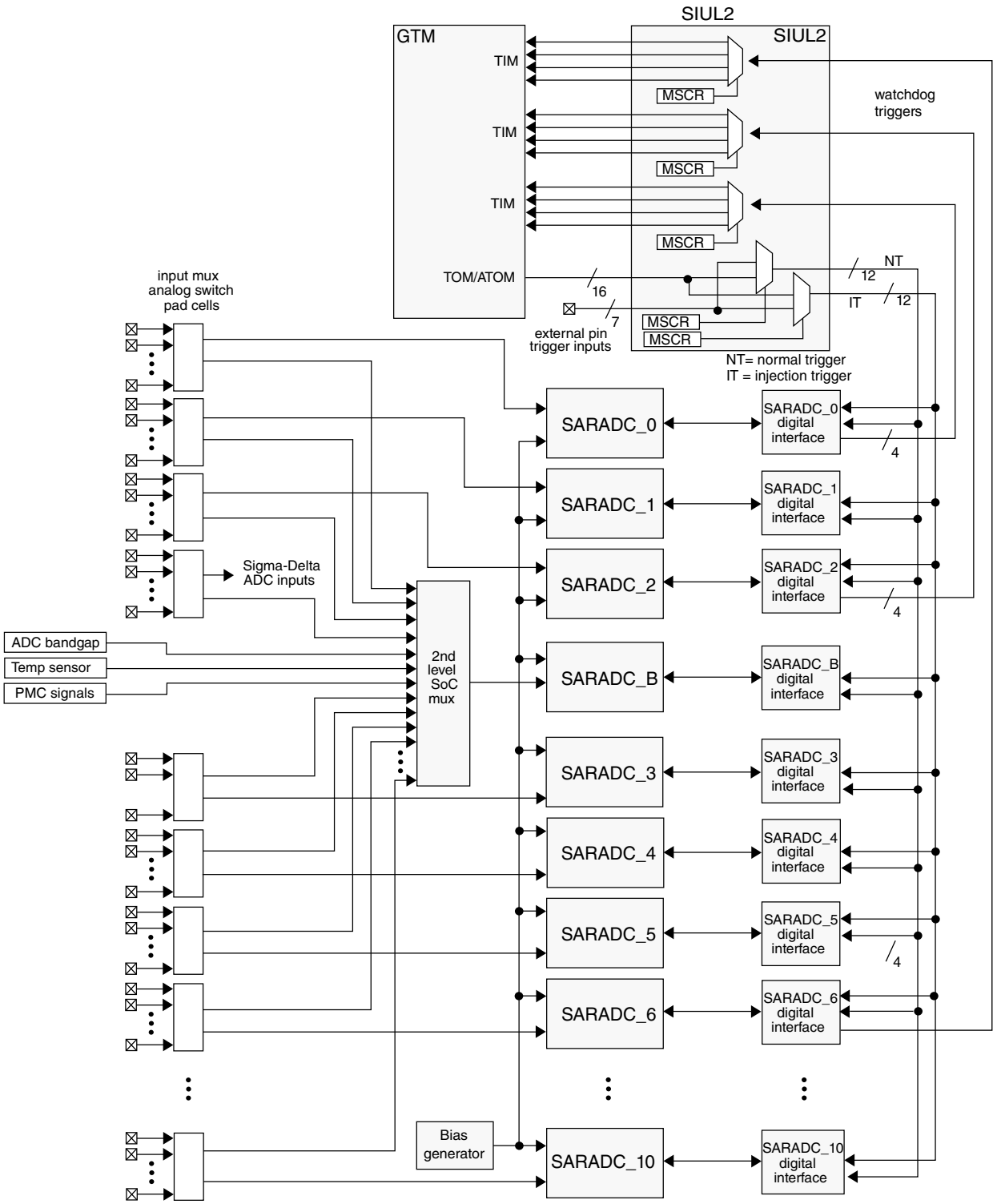


Figure 3-17. SAR ADC integration diagram

3.1.16.2 Self Test features

- The sampling time of SARADC_B was changed from 2.2 ms to 2.5 µs.
  - Extended the sampling time of SARADC_B to 2.5 µs due to long settling time induced by the high impedance of the source (20 kΩ).

3.1.16.3 SARADC register description

- Changed the field name from SARADCx_ICWSELR7 to the following:
  - SARADC8_ICWSELR8
  - SARADC9_ICWSELR9
  - SARADC10_ICWSELR10

3.1.17 FIFO Control register

- Added FRST and FOWEN fields at 15 and 28 bit field in FIFO Control Register.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															0
W																FRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				FTHLD				0				FOWEN	FSIZE		FE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Field	Description
FRST	FIFO Flush Reset This is functional soft reset to flush the current FIFO content and initialize the internal FIFO pointers. This is W1S(write one shot) bit which can be written as '1' but always read as '0'. Use this option only after disabling the ADC i.e. MCR[EN]='0'.

Table continues on the next page...

Field	Description
	0 No effect. 1 Generate a single cycle reset event to flush the FIFO.
FOWEN	FIFO Over Write Enable When set the built-in FIFO structure allows the overwritting of new converted data over older converted data after the FIFO has number of entries equal to the FSIZE. In this case overrun will not be seen. 0 Data FIFO OW option disabled. 1 Data FIFO OW option enabled.

3.1.18 Test Channel Data register

- Added a note 'This register should be accessed 32-bit R/W', in the description of the Test Channel Data register (SARADC_TCDRn).

3.1.19 CAN subsytem

3.1.19.1 CAN Subsystem

- CAN FD support is removed throughout the document. See attached *CAN Subsystem* pdf for details.

3.1.19.2 M_CAN_RXF0C[F0S]

- The width of field M_CAN_RXF0C[F0S] is increased to 7. This field now ranges from bit 9 to bit 15.

3.1.19.3 DRXE

- Added bit field values in the DRXE field.

DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable

	0 Interrupt disabled 1 Interrupt enabled

3.1.20 CTS mode support

The LFAST module supports flow control methods. CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by the proper programming of Rx FIFO Lower threshold (RFCR[RCTSMN]) and Higher threshold (RFCR[RCTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by writing MCR[CTSEN] = 1, then:

- Send the CTS bit as 0 in the LFAST frame (bit[0] of the Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield RFCR[RCTSMX]. This indicates that the LFAST device does not want the peer device to send data. The CTS bit of all frames are sent 0 until the Rx FIFO pointer reaches the lower threshold, as described below.
- Send the CTS bit as 1 in the LFAST frame (bit[0] of the Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield RFCR[RCTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frames are sent 1 until the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in the LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of the Header field are sent as 0011b and 8-bit payload of 0.

If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frames sent will have the CTS bit as 1 in the LFAST frame (bit[0] of the Header field)RCTSMN.

3.1.21 LINFlexD

- In the LIN Status Register, the bit field value description of the LIN field for 0001 Init mode changed to 'LINFlexD is in Initialization mode'.
- In the LIN Control Register2 IOPE field description, changed the 'If generic slave...' sentence to 'If generic slave = 0, then this bit will always read a 0, and cannot be programmed'.

3.1.22 Reset Event Select register

- Updated the register description by adding the sentence, 'When the Temperature Sensor is set to cause Functional Resets, the device will not exit from reset until the temperature has returned to a level that is not detected as an event. Destructive resets from the Temperature Sensor are not affected in the end.

3.1.23 Module Configuration register

- The TCK divide values for TCKSEL field were updated to:
  - 000 TCK is system clock /2
  - 001 TCK is system clock /4
  - 010 TCK is system clock /6
  - 011 TCK is system clock /8
  - 100 TCK is system clock /10
  - 101 TCK is system clock /12
  - 110 TCK is system clock /14
  - 111 TCK is system clock /16

3.1.24 LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug

3.1.24.1 LVLPEN

- Updated the field description of the LVLPEN field in the LFAST LVDS Control register by adding the following sentence:
  - The internal loopback feature is not intended for board level functionality, so this feature should not be implemented.

3.1.24.2 Digrf changed to LFAST

- All the instances of digrf in the Reference Manual and Emulation and Debug Device Reference Manual have been changed to LFAST.
  - pp_digrf_lvds_lpbk_en changed to pp_lfast_lvds_lpbk_en.
  - digrf_sysclk_en and digrf_sysclk changed to lfast_sysclk_en and lfast_sysclk in Figure 76-1 LFAST block diagram and Figure 76-2 LFAST block diagram.

3.1.25 Self-Test Control Unit (STCU2)

3.1.25.1 STCU2

NXP has validated and therefore supports only particular STCU2 built-in self-test (BIST) sequences for this chip. For descriptions of the supported BIST sequences for this chip, see the application note titled *Using the Built-in Self-Test (BIST) on the MPC5777M* (document AN5131).

3.1.25.2 Register write-access watchdog timer

As explained in the STCU2_SKC register description:

- A key mechanism protects STCU2 registers during the self-test configuration phase (both online/offline) by preventing any unwanted access.
- A hardware watchdog timer locks register-write access after a number of STCU2 clock cycles. To refresh the hardware watchdog timer before it times out, write only security key 2 to STCU2_SKC.

AUTOLOCK_VALUE is the number of STCU2 clock cycles after which the hardware watchdog timer locks register-write access. See the chip-specific STCU2 information for the value of AUTOLOCK_VALUE.

Note: For write completion, each DCF record requires a number of STCU2 clock cycles equal to DCF_COMPLETION. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is AUTOLOCK_VALUE divided by DCF_COMPLETION. See the chip-specific STCU2 information for the value of DCF_COMPLETION.

The hardware watchdog timer is particularly useful in case STCU2_CFG[WRP] is 0 during the software self-test configuration. In this case, the software application might enable write access to the STCU2 registers.

3.1.25.3 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion

Register write-access watchdog timer refers to two clock-cycle values related to the STCU2's write-access watchdog. The following table provides these values on this chip.

Clock-cycle value	Number of STCU2 clock cycles
AUTOLOCK_VALUE	1024
DCF_COMPLETION	16

3.1.26 FCCU

- Removed the following note from the FCCU_RFS_CFGn and FCCU_NMI_ENn registers:
  - Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Chapter 4
RM Addendum Rev 4.1

4.1 Overview

The updates for MPC5777M RM Rev 4 is described in detail below.

4.2 UTest flash memory map

In the UTest flash memory map starting at address 0x00400308, the updated “Reserved” row at starting address 0x00400308 now looks like this:

Start address	End address	Allocated size [bytes]	Description	Comments
0x00400308	0x00400347	64	Reserved	Space for 8 DCF records used to load the HSM 'ROM' keys. **Note:** When device is in CUST_DEL (or later) life cycle and prefetching is enabled in the platform flash controller (PFLASH_C55FM), it reads the Reserved locations (0x0400308-0x0400347) and returns dummy DCF records. However, accesses to these locations may result in anomalous operation. Reading the Reserved locations can result in a false ECC event, specifically either a false single-bit correction or a false multi-bit error detection. Reading the Reserved locations can also interfere with an unrelated, but coincident, loading of the PFLASH_C55FM prefetch buffers. As a result, a subsequent request to flash may result in a false ECC event, specifically either a false single-bit correction or a false multi-bit error detection. **Note:** Program MPU to prevent accesses to 0x0400308 - 0x0400347. Avoid accessing the Reserved locations in the flash UTEST region (0x0400308 - 0x0400347). Prior to making access to this region, disable flash buffer prefetches.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number MPC5777MRM
Revision 4.3, 01/2017

MPC5777M Reference Manual

Document Number: MPC5777MRM
Rev. 4, 04/2015

Contents

Section number **Title** **Page**

Chapter 1
Preface

Chapter 2
Introduction

Chapter 3
Embedded memories

Chapter 4
Signal Description

Chapter 5
Memory Map

Chapter 6
Functional Safety

Chapter 7
Device Configuration

Chapter 8
Reset and Boot

Chapter 9
Device Configuration Format (DCF) Records

Chapter 10
Power management

Chapter 11
Security

Chapter 12
Calibration and Debug

Chapter 13
Core Complex Overview

Chapter 14
Core description

Chapter 15
Core description

Chapter 16
System Integration Unit Lite2 (SIUL2)

Chapter 17
Crossbar Switch (XBAR)

Chapter 18
Crossbar Integrity Checker (XBIC)

Chapter 19
Peripheral Bridge (PBRIDGE)

Chapter 20
System Memory Protection Unit (SMPU)

Chapter 21
Intelligent AHB Gasket (IAHBG)

Chapter 22
Semaphores2 (SEMA42)

Chapter 23
Interrupt Controller (INTC)

Chapter 24
Enhanced Direct Memory Access (eDMA)

Chapter 25
Direct Memory Access Multiplexer (DMAMUX)

Chapter 26
Clocking

Chapter 27
Dual PLL Digital Interface (PLLDIG)

Chapter 28
Clock Monitor Unit (CMU)

Chapter 29
Clock Generation Module (MC_CGM)

Chapter 30
OSC Digital Interface (XOSC)

Chapter 31
IRCOSC Digital Interface

Chapter 32
RAM Controller (PRAMC)

Chapter 33
Flash memory controller

Chapter 34
Embedded Flash Memory (c55fmc)

Chapter 35
Decorated Storage Memory Controller (DSMC)

Chapter 36
Flash Memory Programming and Configuration

Chapter 37
External Bus Interface (EBI)

Chapter 38
Analog-to-Digital Converters (ADC) Configuration

Chapter 39
Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface

Chapter 40
Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

Chapter 41
Temperature Sensor

Chapter 42
System Timer Module (STM)

Chapter 43
Software Watchdog Timer (SWT)

Chapter 44
Periodic Interrupt Timer (PIT)

Chapter 45
GTM Integration (GTMINT) Module

Chapter 46
Generic Timer Module (GTM104)

Chapter 47
GTM Development Interface (GTMDI)

Chapter 48
CAN Subsystem

Chapter 49
Deserial Serial Peripheral Interface (DSPI)

Chapter 50
Zipwire

Chapter 51
Serial Interprocessor Interface (SIPI)

Chapter 52
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

Chapter 53
Fast Ethernet Controller (FEC)

Chapter 54
FlexRay Communication Controller (FlexRay)

Chapter 55
Inter-Integrated Circuit (I2C)

Chapter 56
Peripheral Sensor Interface (PSI5)

Chapter 57
Peripheral Sensor Interface-Support Module (PSI5-S)

Chapter 58
SENT Receiver (SRX)

Chapter 59
LINFlexD

Chapter 60
Reset Generation Module (MC_RGM)

Chapter 61
Boot Assist Flash (BAF)

Chapter 62
System Status and Configuration Module (SSCM)

Chapter 63
Power Management Controller digital interface (PMC_dig)

Chapter 64
Power Control Unit (MC_PCU)

Chapter 65
Mode Entry Module (MC_ME)

Chapter 66
Core Debug Support

Chapter 67
Core Debug Support

Chapter 68
Debug and Calibration Interface (DCI)

Chapter 69
JTAG Controller (JTAGC)

Chapter 70
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

Chapter 71
JTAG Data Communication (JDC)

Chapter 72
Sequence Processing Unit (SPU)

Chapter 73
JTAG Master (JTAGM)

Chapter 74
Debug Zipwire

Chapter 75
Debug Serial Interprocessor Interface (SIPI)

Chapter 76
LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug

Chapter 77
Development Trigger Semaphore (DTS)

Chapter 78
Nexus Aurora Router (NAR)

Chapter 79
Nexus Module

Chapter 80
Nexus Module

Chapter 81
Nexus Crossbar Multi-Master Client (NXMC)

Chapter 82
Cyclic Redundancy Check (CRC) Unit

Chapter 83
Memory Error Management Unit (MEMU)

Chapter 84
Indirect Memory Access (IMA)

Chapter 85
Fault Collection and Control Unit (FCCU)

Chapter 86
Self-Test Control Unit (STCU2)

Chapter 87
Register Protection (REG_PROT)

Chapter 88
Password and Device Security Module (PASS)

Chapter 89
Tamper Detection Module (TDM)

Chapter 90
Wakeup Unit WKPU

Chapter 1
Preface

1.1 Overview

The primary objective of this document is to define the functionality of the MPC5777M microcontroller for use by software and hardware developers. The MPC5777M is built on Power Architecture ® technology and integrates technologies that are important for today's and tomorrow's automotive applications. MPC5777M applications include automotive powertrain controller and chassis control for eight cylinder gasoline and diesel engines, transmission control, steering and braking, as well as high end hybrid, electric power steering, chassis, and safety applications that require a high safety integrity level.

The information in this book is subject to change without notice, as described in the disclaimers on the title page. As with any technical documentation, it is the reader's responsibility to be sure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit the Freescale/ Web site at "http://www.freescale.com".

1.2 Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5777M device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

1.3 Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or device-specific information
- Part II includes chapters that describe the functionality of the individual modules on the device
- Part III includes chapters that describe the functionality of the emulation device.

1.4 Register conventions

The following figure and table provide the register conventions used throughout this manual.

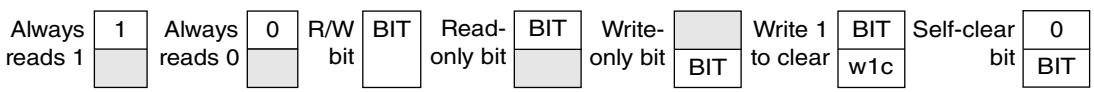


Figure 1-1. Key to register fields

Table 1-1. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register field types	
R	Read only. Writing this bit has no effect.
W	Write only
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset values	
0	Resets to zero
1	Resets to one
—	Undefined at reset
u	Unaffected by reset
[signal_name]	Reset value is determined by polarity of indicated signal.

1.5 Acronyms and abbreviations

The following table lists some acronyms and abbreviations used in this document.

Table 1-2. Acronyms and abbreviations

Term	Meaning
ATX	Analog Test Module
AUTOSAR®	Automotive Open System Architecture
BAF	Boot Assist Flash
CAN	Controller Area Network
CGM	Clock Generation Module
CMU	Clock Monitoring Unit
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
DMAMUX	Direct Memory Access Controller Multiplexer
DSPI	Deserial Serial Peripheral interface
DTS	Development Tool Semaphore
FCCU	Fault Collection and Control Unit
FEC	Fast Ethernet controller
FLEXRAY	Automotive network communications protocol
GPIO	General-purpose I/O
GTM	Generic Timer module
IEEE	Institute for Electrical and Electronics Engineers
IIC	Inter-Integrated Circuit (I^2C)
IMA	Indirect Memory Access
INTC	Interrupt Controller
JDC	JTAG Data Communication
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LFAST	LVDS Fast Asynchronous Serial Transmission
LINFlexD	Local Interconnect Network controller
ME	Mode Entry Module (MC_ME)
MEMU	Memory Error Management Unit
Mux	Multiplex
OSC	Oscillator
PASS	Password and Device Security Module
PCM	Platform Configuration Module
PCU	Power Control Unit
PFLASH	Flash controller
PIT	Periodic Interrupt Timer
PMC	Power Management Controller
PRAM	RAM controller
PSI5	Peripheral Sensor Interface
RCOSC	RC Oscillator
RGM	Reset Generation Module

Table continues on the next page...

Table 1-2. Acronyms and abbreviations (continued)

Term	Meaning
RTL	Register transfer language
Rx	Receive
SD ADC	Sigma Delta Analog to Digital Converter
SAG	Safety Application Guide
SAR ADC	Successive Approximation Register Analog to Digital Converter
SEMA4	Semaphore
SENT	Single Edge Nibble Transmission
SIPI	System Interprocessor Interface
SIUL	System Integration Unit Lite
SMPU	System Memory Protection Unit
SoC	System on Chip
SSCM	System Status and Configuration Module
STCU	Self Test Control Unit
STM	System Timer module
SWT	Software Watchdog Timer
TBD	To be defined
TSENS	Temperature Sensor
TTCAN	Time Triggered CAN
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
UF	Unrecoverable Fault
WKPU	Wakeup Unit
XBAR	Crossbar

1.6 Reference documents

In addition to this reference manual, the following documents provide additional information on the operation of the MPC5777M:

- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture
- Power Architecture Book E V1.0 (http://www.freescale.com/files/32bit/doc/user_guide/BOOK_EUM.pdf)

Chapter 2
Introduction

2.1 MPC5777M microcontroller

The MPC5777M microcontroller is a member of a family of devices superseding the MPC5600 family. The MPC5777M builds on the legacy of the MPC5600 family, while introducing new features coupled with higher throughput to provide a substantial reduction in cost per feature as well as significant improvements in power and performance (MIPS per mW). There are five processor cores on the MPC5777M device.

Table 2-1. Processor cores

Computational Shell		
Core0	Main Core_0	e200z710
Core1	Main Core_1	e200z710
Checker Core0	Checker Core_0s	e200z709
I/O Processor		
Core2	Peripheral Core_2	e200z425
Hardware Security Module		
e200z0h Core		

2.1.1 Target applications

This device is targeted at automotive powertrain controller and chassis control applications for gasoline and diesel engines, transmission control, steering and braking applications, as well as high-end hybrid and advanced combustion systems.

Many of these applications are considered to be functionally safe and are designed to achieve ISO26262 ASIL-D compliance. For this reason safety measures have been added that are further described in Functional Safety chapter as well as the *Safety Manual*.

2.1.2 Overall architecture

The architecture is split into two distinct regions, as shown in Figure 2-1 : the computational shell and the I/O subsystem. The computational shell consists of three CPUs (two in lock-step for safety), fast local memories inside the CPUs, and fast system memory. All components of the computational shell are connected through a fast crossbar switch (XBAR_0).

The I/O subsystems consists of an I/O processor (IOP), DMA, Hardware Security Module (HSM), and all the other peripherals connected through a slow crossbar switch (XBAR_1).

The computational shell is optimized for performance and the I/O subsystem is optimized for low power.

The two crossbars (XBAR_0 and XBAR_1) are connected by a pair of cross-connected master - slave ports. The result of this is that all crossbar masters on both crossbars, can directly access all crossbar slaves on both crossbars. From a software perspective, every bus masters (all CPUs, DMA, SIPI, Ethernet, FlexRay, HSM) sees every memory and peripheral (all RAMs, flash, and peripherals) in a consistent flat memory map.

2.1.3 Core features

The MPC5777M includes four user programmable CPU cores and one safety core. The main computational shell consists of dual e200z7 CPUs operating at up to 300 MHz with a third, identical CPU running as a safety checker core in delayed lockstep mode with one of the dual z7 cores. The I/O subsystem includes a CPU targeted at managing the peripherals. The I/O subsystem core is an e200z4 CPU running at up to 200 MHz and including enhancements to implement Digital Signal Processing (DSP) algorithms. The fifth programmable CPU is an e200z0 running at up to 100 MHz and embedded in the HSM. All CPUs are compatible with the Power Architecture® VLE instruction set, which supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

2.1.4 I/O Processor

The I/O Processor (IOP), also referred to as Peripheral Core_2, is one of five processor cores on the MPC5777M device as shown in Figure 2-1. In general, the intended use of the IOP is to initially configure the machine and then control various peripheral elements of the device. However, the IOP is not limited to any particular duty.

The data bus structure of the IOP, that consists of a 32-bit address bus and a 32-bit data bus, is connected to the slow crossbar switch, which operates at 100 MHz. The instruction bus structure of the IOP, which consists of a 32-bit address bus and a 32-bit data bus, is connected to the fast crossbar switch, which operates at 200 MHz. The address bus structure and the data bus structure of Main Core_0 and Main Core_1 are also connected to the fast crossbar switch. The data bus structure of the IOP is connected to the slow crossbar switch, which operates at 100 MHz. Thus, data fetches run at only one-half the speed of instruction fetches.

It is possible to fetch data from memory elements connected to the fast crossbar switch and it is also possible to fetch instructions from memory elements connected to the slow crossbar switch. In both cases, the instruction and data paths would have to go through both the slow and fast crossbar switches and would be relatively slow. When the IOP first starts, it will start executing from memory elements on the fast crossbar switch. However, good programming techniques will cause the IOP to fill its internal instruction cache so that accesses through the fast crossbar switch occur less often.

The I/O Processor is automatically started during the boot-up sequence after the release of reset. This process is fully described in Reset and Boot chapter. During the boot sequence the System Status and Configuration Module (SSCM) reads and interprets a set of device configuration records located in the TEST flash area. One of these records is the start address of the code where the IOP will begin program execution. This vector will always point to a location in the Boot Assist Flash (BAF). Once the SSCM finds and interprets this particular device configuration record, it will write the start address for the IOP to the appropriate location in the Mode Entry Module (MC_ME). When the internal reset signal is released, the MC_ME transfers the IOP start address to the IOP and code execution begins at that address.

The code for the Boot Assist Flash, the TEST Device Configuration records and some of the UTEST Device Configuration records are written by the device manufacturer and are programmed into the device during production testing. All MPC5777M devices are shipped with startup code programmed into the BAF. Also, the MPC5777M is configured such that code execution by the IOP must begin with a start address located in the BAF.

If there is no user application code for the IOP to execute, the IOP, executing pre-programmed code in the BAF, will attempt to download user application code through the LINFlexD port using a specific data transfer protocol. Serial download can also be initiated from the CAN. This downloaded code is installed into program memory and code execution will begin at the start address embedded in the downloaded code.

As previously stated, the IOP's data bus structure is connected to the slow crossbar switch, which provides access to many of the peripheral devices on the device. The instruction bus of the IOP is connected to the fast crossbar switch, which gives the IOP access to the BAF, as well as all of the other memory elements. When the IOP begins

code execution, a considerable amount of traffic through the fast crossbar switch will be required. The IOP has both internal instruction cache and internal data cache. Once the IOP gets started and the internal instruction cache begins to fill, significantly less traffic will be required by the IOP through the fast crossbar switch. Accesses to the peripheral elements on the slow crossbar switch is transparent to activity occurring on the fast crossbar switch.

The IOP is placed in the MPC5777M architecture in such a way that enables it to access many of the peripheral elements without affecting the processing speed of Main Core_0 and Main Core_1. On the MPC5777M device, there are several analog-to-digital channels taking data that must be analyzed for various spectral components.

The IOP has an integrated DSP processor that has special hardware to execute multiply and accumulate operations. Because the DSP unit functions as a coprocessor integrated into the e200z425 core, data fetched by the z425 core can be evaluated with much greater speed than if the data had to reside in a memory element outside the z425 core.

One intended purpose for the IOP is to manage a system of analog-to-digital converters and fetch the digital data that is generated. The integrated DSP unit uses a signal processing algorithm to digest the incoming data. The results of this signal processing algorithm are used in other engine control algorithms. Using the IOP to manage the analog-to-digital converters and do the initial signal processing of the data allows Main Core_0 and Main Core_1 to focus on other compute intensive applications instead of managing relatively slow peripheral elements.

Once the IOP has performed its initialization duties (as determined by pre-loaded instruction code in the BAF) and downloaded any needed applications code, the IOP can essentially be programmed to perform any duty or access any memory element. When executing instructions from its internal instruction cache and fetching data from peripherals connected to the slow crossbar switch, the IOP operates autonomously from Main Core_0 and Main Core_1.

2.1.5 Memory hierarchy

The MPC5777M computational shell has three levels of memory hierarchy. At the top level is a 16 KB instruction cache and a 4 KB data cache; 16 KB local instruction RAM; 64 KB local data RAM. The local RAM arrays are embedded into the CPU's pipeline to provide the fastest possible access time. The second level of memory is system RAM and flash memory. These are connected by a 200 MHz, 64-bit wide crossbar. The third level of memory system is the I/O subsystem. All of the I/O peripherals are connected together

by a 100 MHz, 64-bit wide crossbar switch, while peripherals are connected at either 100 or 50 MHz, depending on their need. The reduced speed for the peripherals helps to reduce power.

2.2 Features summary

On-chip modules within MPC5777M include the following features:

- Three main CPUs, dual issue, 32-bit CPU core complexes (e200z7), one of which is a dedicated lockstep core.
  - Power Architecture embedded specification compliance
  - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
  - Single-precision floating point operations
  - 16 KB Local instruction RAM and 64 KB local data RAM
  - 16 KB I-Cache and 4 KB D-Cache
- I/O Processor, dual issue, 32-bit CPU core complexes (e200z4), with
  - Power Architecture embedded specification compliance
  - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
  - Single-precision floating point operations
  - Lightweight Signal Processing Auxiliary Processing Unit (LSP APU) instruction support for digital signal processing (DSP)
  - 16 KB Local instruction RAM and 64 KB local data RAM
  - 8 KB I-Cache
- 8640 KB on-chip flash
  - Supports read during program and erase operations, and multiple blocks allowing EEPROM emulation
- 404 KB on-chip general-purpose SRAM including 64 KB standby RAM (+ 192 KB data RAM included in the CPUs). Of this 404 KB, 64 KB can be powered by a separate supply so the contents of this portion can be preserved when the main MCU is powered down.

- Multichannel direct memory access controllers (eDMA): 2 x 64 channels per eDMA (128 channels total)
- Triple Interrupt controller (INTC)
- Dual phase-locked loops with stable clock domain for peripherals and FM modulation domain for computational shell
- Dual crossbar switch architecture for concurrent access to peripherals, flash, or RAM from multiple bus masters with end-to-end ECC
- Hardware Security Module (HSM) to provide robust integrity checking of flash memory
- System Integration Unit Lite (SIUL)
- Boot Assist Module (BAM) supports factory programming using serial bootload through 'UART Serial Boot Mode Protocol'. Physical interface (PHY) can be:
  - UART/LIN
  - CAN
  - FlexRay
- GTM104 — generic timer module
- Enhanced analog-to-digital converter system with
  - Twelve separate 12-bit SAR analog converters
  - Ten separate 16-bit Sigma-Delta analog converters
- Eight deserial serial peripheral interface (DSPI) modules
- Two Peripheral Sensor Interface (PSI5) controllers
- Three LIN and three UART communication interface (LINFlexD) modules (6 total)
  - LINFlexD_0 is a Master/Slave
  - LINFlexD_1, LINFlexD_2, LINFlexD_14, LINFlexD_15, and LINFlexD_16 are Masters
- Four modular controller area network (MCAN) modules and one time-triggered controller area network (M-TTCAN)
- External Bus Interface (EBI)
  - Dual routing of accesses to EBI

- Access path determined by access address
- Access path downstream of PFLASH controller
  - Allows EBI accesses to share buffer and prefetch capabilities of internal flash
  - Allows internal flash accesses to be remapped to memories connected to EBI
- Access path via dedicated AXBS slave port
  - Avoids contention with other memory accesses

- Two Dual-channel FlexRay controllers
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)
- Self-test capability

2.3 Feature list

The following table lists a summary of major features for the MPC5777M device. The feature column represents a combination of module names and capabilities of certain modules. A detailed description of the functionality provided by each on-chip module is given later in this document.

Table 2-2. Features List

Feature	Description
MPC5700 family	55 nm process
Main Cores	
Main Core	e200z7
Number of Main Cores	2
Number of checker cores	1
Local RAM (per main core)	16 KB Instruction 64 KB Data
Single Precision Floating Point	Yes
VLE	Yes
Cache	16 KB Instruction 4 KB Data
Peripheral Core	

Table continues on the next page...

Table 2-2. Features List (continued)

Feature	Description
Peripheral Core	e200z4
Local RAM	16 KB Instruction 64 KB Data
Single Precision Floating Point	Yes
SIMD	Yes
VLE	Yes
Cache	8 KB Instruction
Other	
MMU Entries	0
MPU	Yes
Semaphores	Yes
CRC Channels	2
Software Watchdog Timer (SWaT)	4
Core Nexus Class	3+
Sequence Processing Unit (SPU)	16
Debug and Calibration Interface (DCI) / Run control Module	Yes
System SRAM	404 KB (Including 64 KB standby)
Flash	8640 KB
Flash fetch accelerator	Two 64-bit AHB ports, 256-bit read data interface. Each AHB port contains 4-entry, 2-way set-associative mini-cache
Data flash (EEPROM)	8 × 64 KB
Flash Overlay RAM	16 KB
External bus	Yes
Calibration Interface	64-bit IPS Slave
DMA channels	2 × 64
DMA Nexus Class	3
LINFlexD	6
M_CAN	4
M_TTCAN	1
DSPI	8
Microsecond channel downlink	Yes
SENT bus	15
I^2C	2
PSI5 bus	5
FlexRay	2 × Dual channel
Ethernet	Yes

Table continues on the next page...

Table 2-2. Features List (continued)

Feature	Description
SIPI / LFAST Interprocessor bus	High Speed
System Timers	8 PIT chan 4 AUTOSAR® (STM)
GTM Timer	48 Input Channels, 152 Output Channels
GTM RAM	54 KB
Interrupt controller	722 sources
ADC (SAR)	12
ADC (SD)	10
Temp. sensor	Yes
PLL	Dual PLL with FM
External Power Supplies	5 V 3.3 V 1.2 V
Low Power Modes	Stop Mode Slow Mode

2.4 Packaging

The MPC5777M is available in the following production packages: 416 BGA and 512 BGA. All BGA packages support development and production applications with the same package.

2.5 Software debug and calibration

The Production Device (PD) includes many features to facilitate software development and calibration of constant data. These features include:

- Nexus code execution trace on all CPUs
- Nexus data trace on bus masters; DMA, Interprocessor bus, CPUs, and so on. Note that the Buddy Device—LFAST does not include Nexus data trace.
- Performance monitors for each CPU
- Sequence Processing Unit for complex trace triggering
- 16 KB of overlay RAM

For more complex software development and calibration of constant data, a Buddy Device (BD) is packaged together with the MPC5777M production device as a multi-die emulation and is referred to as an Emulation Device (ED).

The ED provides exactly the same features as the PD. The ED is available in the same package as the PD but has an additional overlay RAM (up to 1M byte in MPC5777M) and a high-speed Nexus trace output port consisting of four lanes of Aurora (LVDS) operating up to 1.25 Gbit/s.

2.6 Block diagrams

The figures below show the top-level block diagrams.

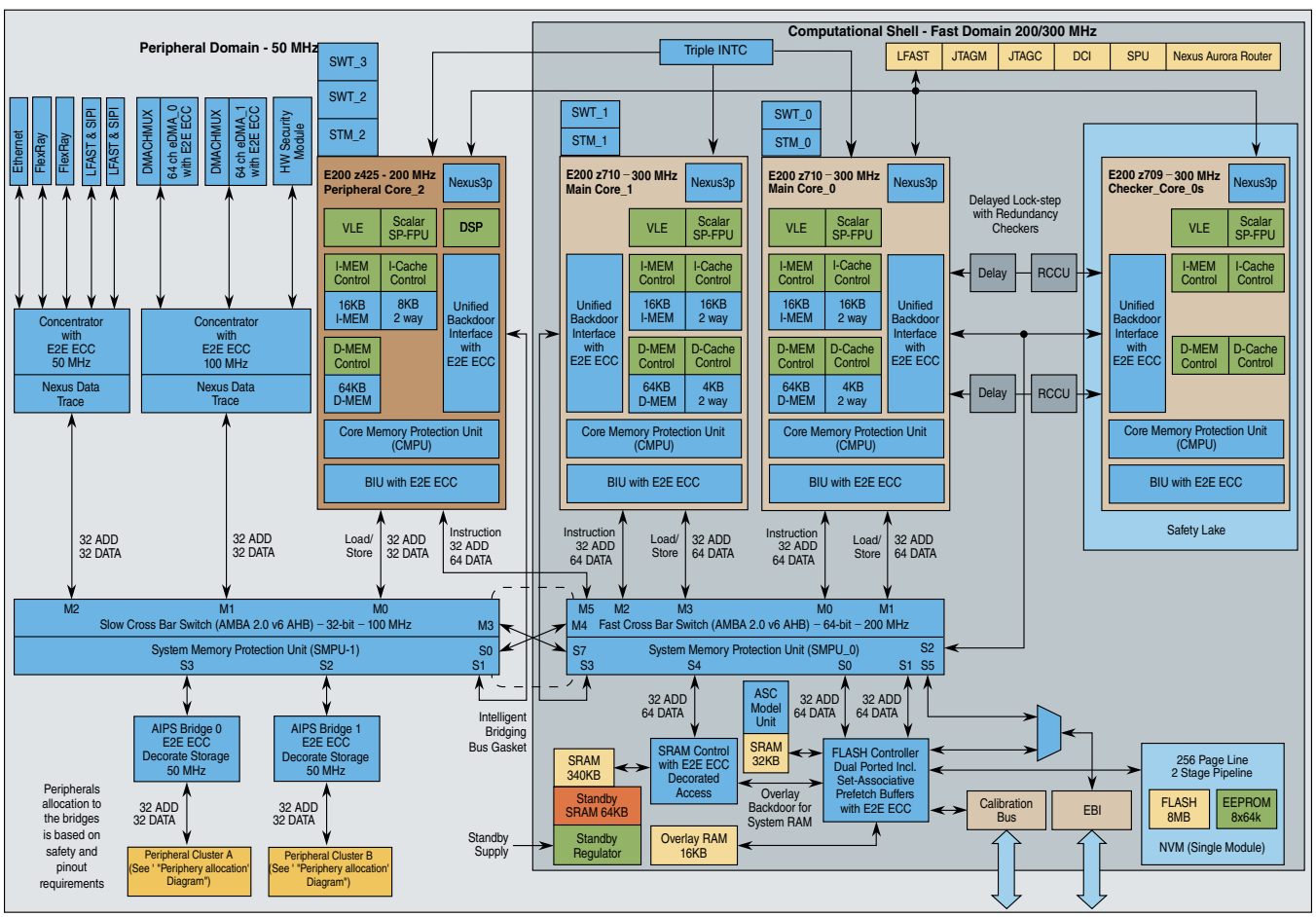


Figure 2-1. Block diagram

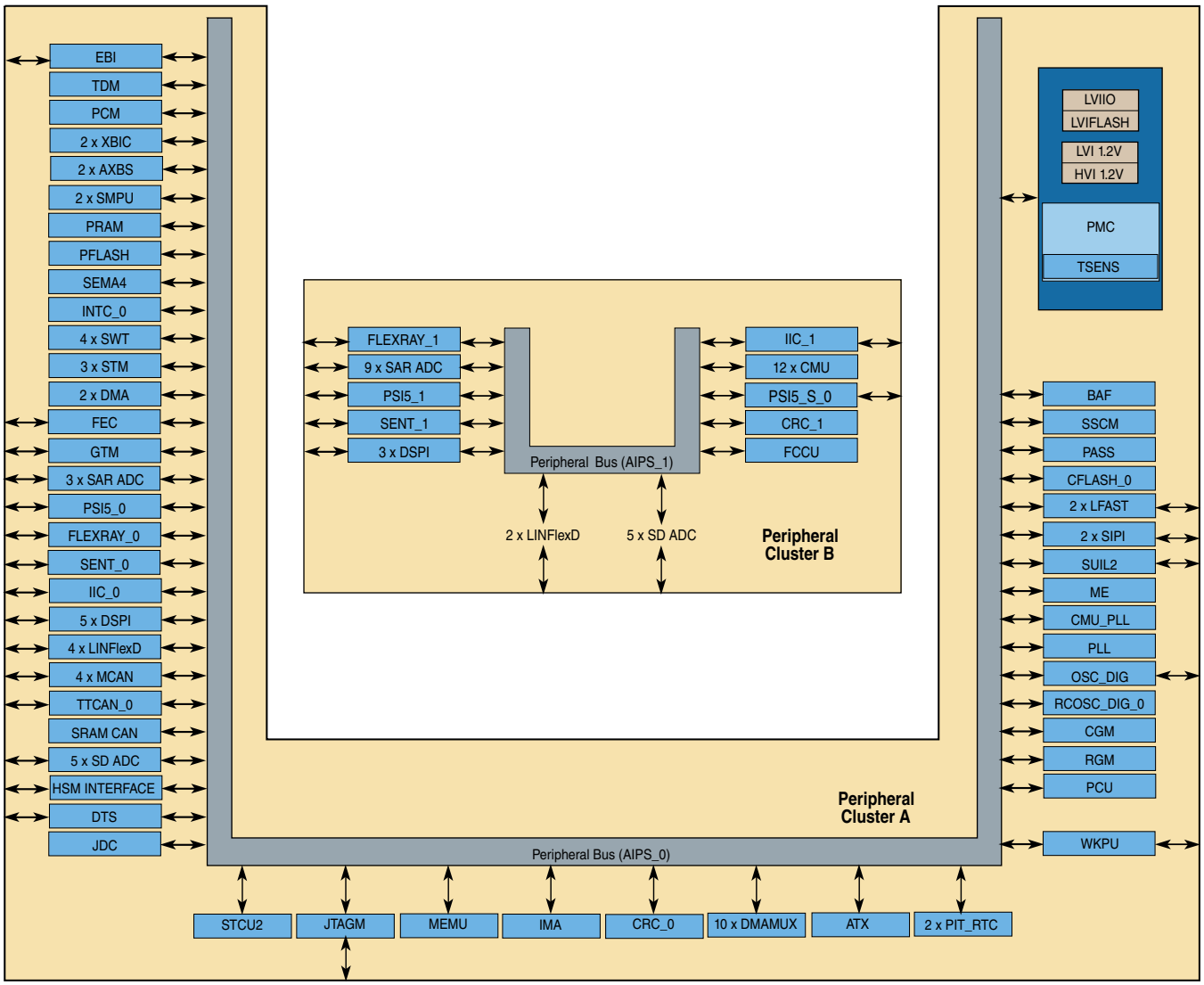


Figure 2-2. Periphery allocation

Chapter 3
Embedded memories

3.1 Overview

The embedded memory architecture for MPC5777M devices is a significant departure from previous-generation devices. Features include:

- Onboard SRAM, including system SRAM, local instruction and data memory for each processor core, and overlay SRAM
- Onboard system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Built-in flash memory security features including censorship and a programmable Hardware Security Module (HSM)
- Embedded memories in the peripherals[1] :
  - CAN
  - eDMA
  - FEC
  - FlexRay
  - GTM

3.2 SRAM

This section describes SRAM.

1. Refer to the respective module chapter for details.

3.2.1 System SRAM

MPC5777M devices include 404 KB of general-purpose on-chip ECC SRAM, including 64 KB standby RAM. The SRAM can be configured for either zero- or one-wait state read operation latency using the PFCR1 register in the SRAM controller.

- See RAM controller (PRAM) chapter for details on configuring the SRAM controller.
- See System RAM memory maps for the MPC5777M SRAM map.

3.2.2 Standby SRAM Regulator

A total of 64 KB of the system SRAM on the device can remain powered when the rest of the device is powered down. This is achieved with the dedicated VDDSTBY pin in the packages. When the standby RAM is powered by VDDSTBY, the RAM is not operational, but the contents are retained.

The Standby RAM regulator has the following key features:

- Automatic, safe, switch between operational (VDD_LV) and standby (VDDSTBY) supplies
- Internal regulation of the high voltage VDDSTBY supply to the low voltage standby RAM supply
- Brownout detection (minimum RAM data retention voltage violation) and reporting via a S/W clearable register bit

The Standby RAM regulator block diagram is given in the following figure.

There are two primary modes of operation with the standby RAM feature:

- Standby RAM feature is not required in the application - in this case, the VDDSTBY pin is connected to ground, and the standby RAM on the device is supplied by VDD_LV and will behave as normal RAM for all conditions.
- Standby RAM feature is required in the application - in this case, the VDDSTBY pin is connected to a voltage within the range given in the device data sheet. The VDDSTBY pin is assumed to be connected to a constant supply, and present before and after the power up/down of the device LV and HV supplies. When the VDD_LV supply to the RAM is above the LV operational voltage threshold, the standby RAM is fully operational and behaves exactly the same as the non-standby system RAM.

When the VDD_LV supply falls below the LV Detect threshold, the RAM low voltage supply is switched to the standby supply. For voltages above 1.2V, the standby regulator is enabled. For voltages below 1.2V, the VDDSTBY pin voltage is passed through directly to the RAM. See the device data sheet for min/max VDDSTBY values, LV supply thresholds, and the operational and brownout detection voltages.

The brownout detection feature of the standby RAM regulator informs the user if any voltage dip caused a RAM retention failure in standby mode. The brownout detector monitors the supply to the RAM array, and sets a latch in the PMC digital interface, when the voltage falls below the minimum RAM retention voltage given in the device data sheet. The latch is set on power up of the device. S/W then clears the latch by writing a '1' to the SBRAM_BRDET bit in the VD_UTST register of the PMC digital interface. If the minimum RAM retention voltage is violated, the latch will be set again. The latch is set on every power-up of VDD_LV unless VDDSTBY is present and within the specified ranges in the data sheet. See Standby RAM Regulator Interface Logic, for more information on the VDDSTBY pin.

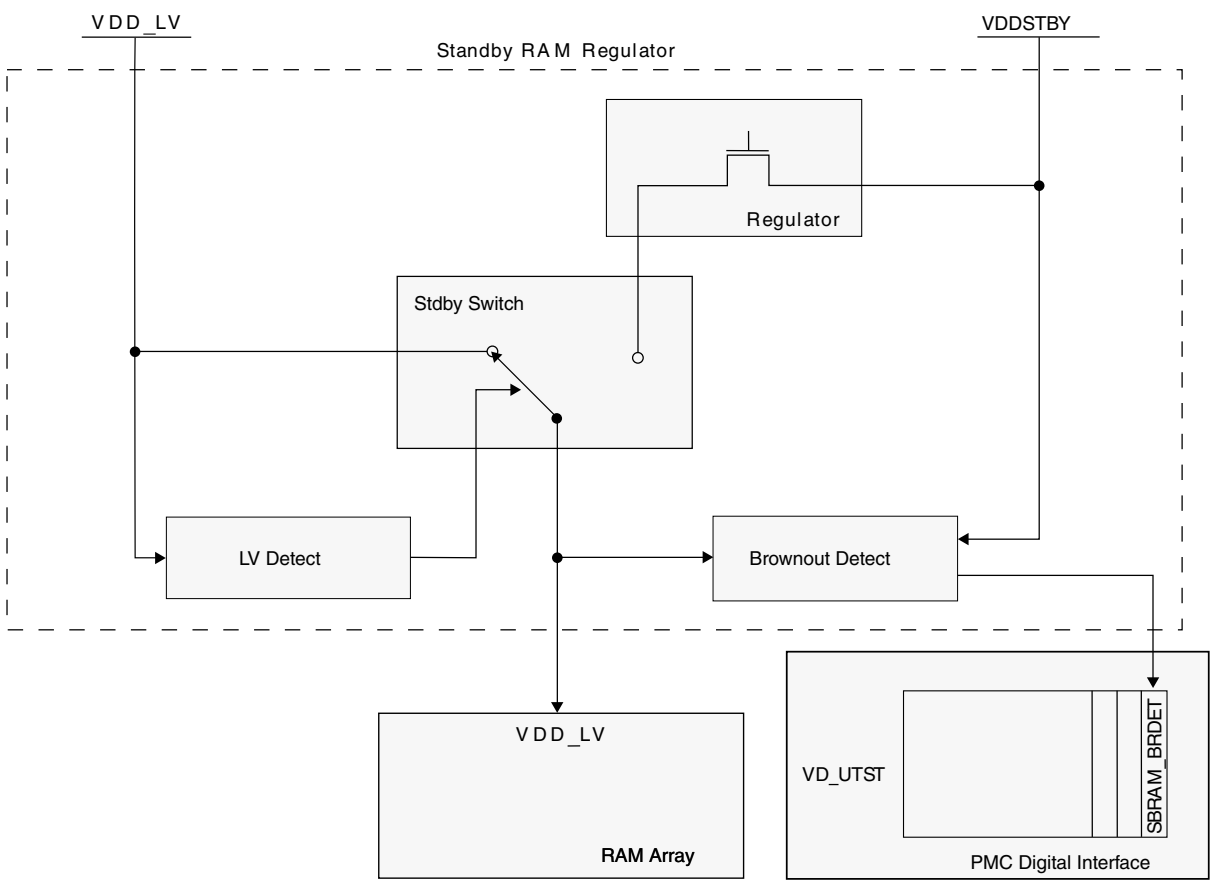


Figure 3-1. Standby RAM Regulator Block Diagram

3.2.3 Overlay SRAM

Overlay SRAM is included in the MPC5777M family of devices as part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables the calibration of constant data without requiring additional external RAMs and calibration memory interfaces.The MPC5777M has two 8 KB banks of ECC-protected RAM, for a total of 16 KB.

Note

Overlay SRAM is normally used for system development purposes only. Code can be executed from internal overlay RAM and extended overlay RAM where available. Execution from overlay RAM remapped over Flash can be enabled or disabled by use of PFCRCR[IRMEM]. Code can be executed directly from Extended Overlay RAM (address range beginning 0x0C000000) on the ED device or internal Overlay RAM (address range beginning 0x0D000000) regardless of the PFCRCR[IRMEM] setting.

Overlay memory can also be used to hold the device debug trace stream without requiring complex debug tool hardware and access to a high-speed trace port. Timing for calibration accesses to a particular overlay SRAM is the same as access to the underlying flash memory if the overlay SRAM is not being used as a destination for trace streaming at the same time.

The calibration remap function supports three different types of overlay SRAM:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is available only on special emulation and debug devices.
- A portion of the system RAM can be used as the overlay RAM.

The overlay function is implemented using registers in the flash memory controller.

1. Using PFlash Calibration Region Descriptors (PFCRDn) define one or more (up to 32) overlay regions.
2. Enable individual regions using the appropriate bits in the PFLASH Remap Descriptor Enable Register (PFDCRDE).

3. Enable overlay using the PFCRCR[GRMEN] field.

See PFlash calibration remap support section, for details on memory overlay (remapping).

3.2.4 Processor core local SRAM

MPC5777M devices include local instruction and data SRAM (I-MEM and D-MEM) for each processor core to provide enhanced performance.

- Each main processor core has 16 KB of instruction SRAM and 64 KB of data SRAM.
- The peripheral processor core has 16 KB of instruction SRAM and 64 KB data SRAM.

The core SRAM supports zero wait-state, single-cycle latency on processor local accesses.

Each area of core SRAM is accessible by other processor cores and bus mastering peripheral devices. See System RAM memory maps for details.

3.3 Embedded flash memory

Flash memory on MPC5777M devices consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. The following figure shows the memory architecture.

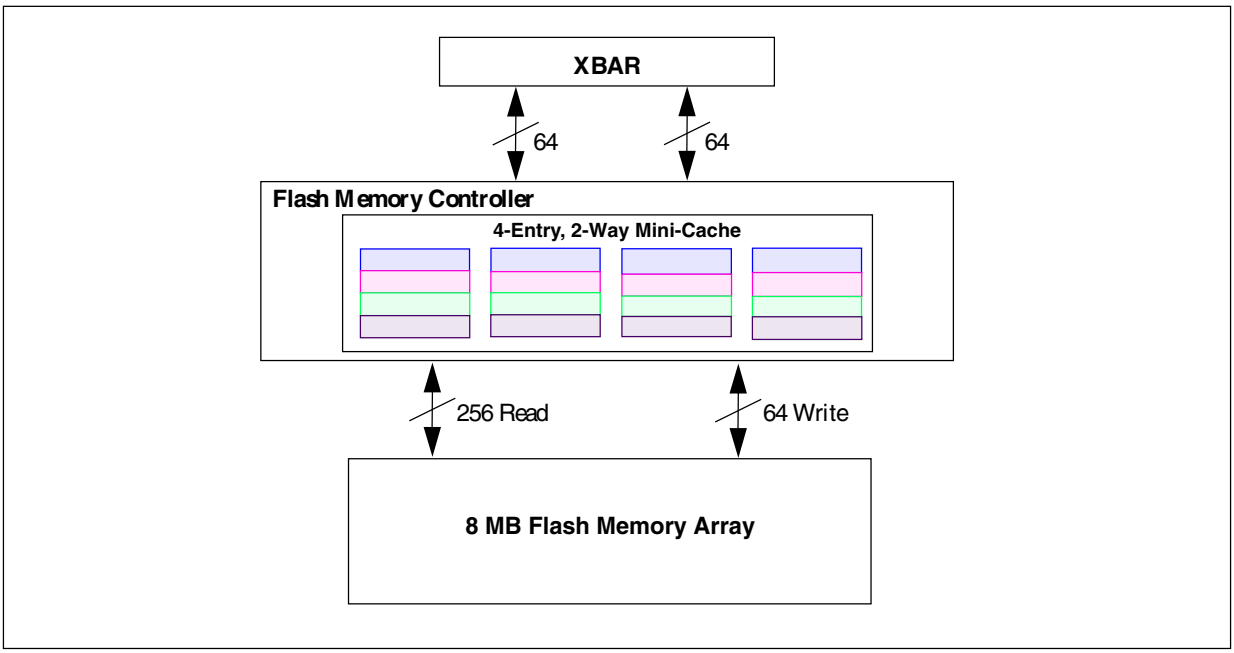


Figure 3-2. MPC5777M flash memory block diagram

The remaining sections give the functional details of the flash controller and flash array, followed by the memory maps.

3.3.1 Flash memory controller

The MPC5777M flash memory controller acts as an interface between the system bus and the flash memory array and serves as the interface to the on-chip overlay RAM and off-chip Buddy Device.[2]

The flash memory controller contains a four-entry, two-way set-associative mini-cache that delivers flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one flash page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed flash memory array access.

The flash memory controller contains configuration registers that manage flash functionality such as read buffering in the mini-cache, access control, calibration RAM overlay remapping, and read wait state management of the flash memory.

See Flash memory controller chapter for details.

2. The Buddy Device is an optional companion chip with additional memory and debug features. A Buddy Device is combined with a Production Device to form an Emulation and Debug Device (ED) for use during system development. It is not used in production systems.

3.3.2 Flash memory array

MPC5777M devices include a single 8640 KB flash memory array, referred to as "main space," plus an independent 16 KB block of one-time-programmable (OTP) flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 16 KB block is referred to as "UTEST space."

Reads of the embedded flash memory return a 256-bit page of data that may be buffered in the mini-cache in the flash memory controller. Programming of the flash may be done by double word (64-bits), page (256-bits), or quad-page (1024-bits). Flash memory is erased on a block by block basis.

3.3.2.1 Features

- Flash segmentation
  - 16 KB, 32 KB, 64 KB, and 256 KB blocks provided for code or data
  - Eight 64 KB blocks provided for EEPROM emulation
  - One 16 KB block and two 64 KB block provided for secure code
  - Two 16 KB blocks provided for secure data
  - Support for reading-while-writing when the accesses are to different partitions
- Flash memory protection: write protection and OTP function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTEST block
- ECC with double bit detection, single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.3.2.2 Flash organization

The embedded flash memory consists of six address spaces in four groupings:

- 16 KB low address space, 32 KB low address space, and 64 KB low address space
- 16 KB mid address space
- 64 KB high address space
- 256 KB address space

Address space group (low-, mid-, or high-address space) determines which registers/ fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The flash module is further divided into ten partitions that determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a "write" (program or erase) to a given partition, it can simultaneously read from any other partition.

- For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written).
- For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The following figure shows the flash block segmentation and read-while-write partitioning for MPC5777M devices.

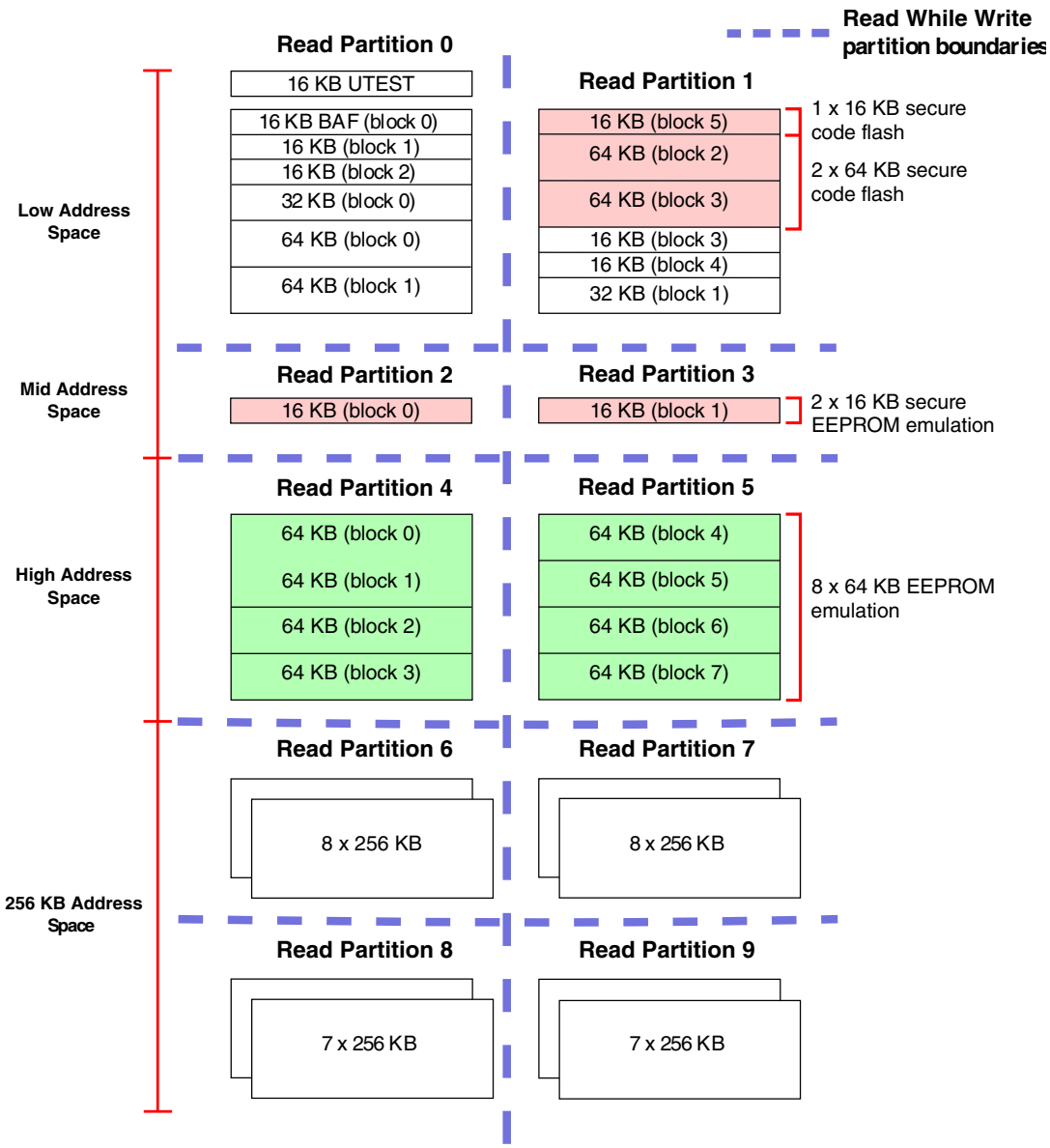


Figure 3-3. Flash memory segmentation and read-while-write partitioning

Note

See Table 5-3 in Memory Map" for the MPC5777M flash memory map.

3.3.2.3 UTEST memory space

Devices in the MPC5777M family contain a 16 KB area of One-Time Programmable (OTP) flash memory for storing test information and device configuration data. See Memory map tables in Memory Map" for the MPC5777M UTEST flash memory map.

3.3.2.4 Chip-specific C55FMC flash register information

The following sections contain chip-specific layouts of the on-chip flash memory control registers that contain bits used to lock individual flash blocks from accidental erase or select individual flash blocks for operations.

3.3.2.4.1 C55FMC_LOCK0 register bit mapping

The flash memory blocks mapped to the C55FMC_LOCK0 register bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	TSLOCK	TSLOCK_ALT	LOWLOCK[13:0]														MIDLOCK[15:0]															
Flash Block Name	UTest NVM Block Space 16 KB	UTest NVM Block Space 16 KB Alt. Interface	—	—	64 KB HSM Code block3				64 KB HSM Code block2				64 KB Code Flash block1				64 KB Code Flash block0				32 KB Code Flash block1				32 KB Code Flash block0				16 KB HSM Code block5			
	16 KB Code Flash block4								16 KB Code Flash block3				16 KB Code Flash block2				16 KB Code Flash block1				16 KB BAF block0				—	—	—	—	—	—	—	—
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB HSM EEPROM block1				16 KB HSM EEPROM block0							

3.3.2.4.2 C55FMC_LOCK1 register bit mapping

The flash memory blocks mapped to the C55FMC_LOCK1 register bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Register Field	Reserved															HIGHLOCK[15:0]							
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Register Field	A256KLOCK[31:0]																							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Flash Block Name	—	—	256 KB Code Flash block29	256 KB Code Flash block28	256 KB Code Flash block27	256 KB Code Flash block26	256 KB Code Flash block25	256 KB Code Flash block24	256 KB Code Flash block23	256 KB Code Flash block22	256 KB Code Flash block21	256 KB Code Flash block20	256 KB Code Flash block19	256 KB Code Flash block18	256 KB Code Flash block17	256 KB Code Flash block16	256 KB Code Flash block15	256 KB Code Flash block14	256 KB Code Flash block13	256 KB Code Flash block12	256 KB Code Flash block11	256 KB Code Flash block10	256 KB Code Flash block9	256 KB Code Flash block8

	24	25	26	27	28	29	30	31
Flash Block Name	256 KB Code Flash block7	256 KB Code Flash block6	256 KB Code Flash block5	256 KB Code Flash block4	256 KB Code Flash block3	256 KB Code Flash block2	256 KB Code Flash block1	256 KB Code Flash block0

3.3.2.4.3 C55FMC_LOCK2 register bit mapping

The flash memory blocks mapped to the C55FMC_LOCK2 register bits are as follows.

3.3.2.4.4 C55FMC_LOCK3 register bit mapping

The flash memory blocks mapped to the C55FMC_LOCK3 register bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Register Bit Name	Reserved																A256KLOCK[47:32]															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

3.3.2.4.5 C55FMC_SEL0 register bit mapping

The flash memory blocks mapped to the C55FMC_SEL0 register bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	TSLOCK	TSLOCK_ALT	LOWSEL[13:0]														MIDSEL[15:0]															
Flash Block Name	UTest NVM Block Space 16 KB	UTest NVM Block Space 16 KB Alt. Interface	—	—	64 KB HSM Code block3	64 KB HSM Code block2	64 KB Code Flash block1	64 KB Code Flash block0	32 KB Code Flash block1	32 KB Code Flash block0	16 KB HSM Code block5	16 KB Code Flash block4	16 KB Code Flash block3	16 KB Code Flash block2	16 KB Code Flash block1	16 KB BAF block0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB HSM EEPROM block1	16 KB HSM EEPROM block0

3.3.2.4.6 C55FMC_SEL1 register bit mapping

The flash memory blocks mapped to the C55FMC_SEL1 register bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																HIGHSEL[15:0]															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	FMC EEPROM Block7	FMC EEPROM Block6	FMC EEPROM Block5	FMC EEPROM Block4	FMC EEPROM Block3	FMC EEPROM Block2	FMC EEPROM Block1	FMC EEPROM Block0

3.3.2.4.7 C55FMC_SEL2 register bit mapping

The flash memory blocks mapped to the C55FMC_LOCK2 register bits are as follows.

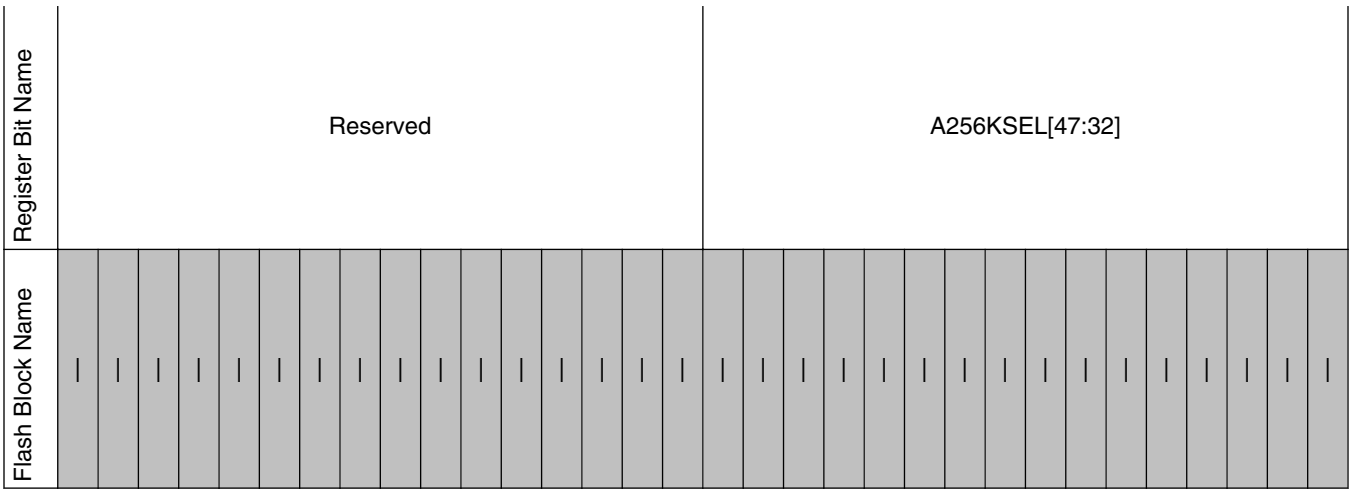
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	A256KLOCK[31:0]																															
Flash Block Name	—	—	256 KB Code Flash block29	256 KB Code Flash block28	256 KB Code Flash block27	256 KB Code Flash block26	256 KB Code Flash block25	256 KB Code Flash block24	256 KB Code Flash block23	256 KB Code Flash block22	256 KB Code Flash block21	256 KB Code Flash block20	256 KB Code Flash block19	256 KB Code Flash block18	256 KB Code Flash block17	256 KB Code Flash block16	256 KB Code Flash block15	256 KB Code Flash block14	256 KB Code Flash block13	256 KB Code Flash block12	256 KB Code Flash block11	256 KB Code Flash block10	256 KB Code Flash block9	256 KB Code Flash block8	256 KB Code Flash block7	256 KB Code Flash block6	256 KB Code Flash block5	256 KB Code Flash block4	256 KB Code Flash block3	256 KB Code Flash block2	256 KB Code Flash block1	256 KB Code Flash block0

3.3.2.4.8 C55FMC_SEL3 register bit mapping

The flash memory blocks mapped to the C55FMC_SEL3 register bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...



3.4 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC (e2eECC) support. This e2eECC is structurally different to traditional "ECC at memory" functionality because it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information storage in one or more intermediate components.

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors that have occurred on data stored in the memory. On the other hand, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master and performs a data integrity check using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated on the basis of more than just the data associated with a storage location to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnect or in the memory unit. The error protection codes may be checked at the memory unit on a store to ensure that no address information or data has been corrupted while the request

has transitioned through the device from the bus master source. The error protection codes may also be checked to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way) or it may simply store the received data and protection codes at the address it receives.

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure) and the stored data was error free, to the extent the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in the following table.

Table 3-1. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields for the computational shell or 32-bit data field for the peripheral shell
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations "covered" by the ECC checks is readily apparent with the e2eECC concept providing improved fault detection capabilities in two important aspects:

- The entire data transaction; from the initiating bus master, through the entire platform crossbar steering mechanism, to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
- The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the checkbit decode and error syndrome generation with the error/no_error state, affecting whether the system bus transfer must be stalled (to correct a single bit error or report a noncorrectable event) or is allowed to complete (for error-free transfers) in a critical timing arc which often sets the upper limit of the operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic; producing an improved timing arc and generally higher operating speeds.

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requester. While not all possible errors can be detected this way, this approach provides a substantial improvement versus traditional methods. Additional on-line diagnostics, or additional hardware, or both should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

3.5 Security features

The MPC5777M architecture uses a combination of device life cycle monitoring and password protection to limit the conditions under which debugging interfaces can access certain areas of flash memory.

Note

Detailed information on the MPC5777M security architecture is published in a separate document and is available on a limited basis to customers who have a non-disclosure agreement (NDA) with Freescale Semiconductor Inc.

Chapter 4
Signal Description

4.1 Production packages

The following production packages are available for the MPC5777M device:

- 416 BGA
- 512 BGA

The 512 BGA is a superset of the 416 BGA. Compared to the 416 BGA, the 512 BGA contains additional GPIO and analog inputs.

The EBI (External Bus Interface) is the only new external interface on MPC5777M that is not currently on other family devices. The EBI is supported on the 512 BGA package but not on the 416 BGA package.

Case number and outline drawings for each package are provided in the MPC5777M data sheet.

4.2 Emulation packages

The following emulation packages are available for the MPC5777M device

- 416 BGA: The emulation and production devices are maintained in the same 416 BGA footprint. The emulation signals are bonded out in the emulation BGA, and not in the production BGA The JTAG interface on the buddy device is bonded out to the JTAG pins in the emulation package. The JTAG interface on the production device is bonded out to the JTAG pins in the production package.
- 512 BGA: The emulation and production devices are maintained in the same 512 BGA footprint.

The case outline drawings for the emulation packages are provided in the MPC5777M data sheet.

4.3 Package pinouts and ballouts

For pin package pinouts and ballouts refer to the MPC5777M Microcontroller Data Sheet.

4.4 Pin/ball descriptions

The complete Signal Description of the MPC5777M device is provided the following sections:

- Pin/ball startup and reset states
- Power supply and reference voltage pins/balls
- System pins/balls
- LVDS pins/balls
- Generic pins/balls

In addition, the signal multiplexing options are provided in two registers:

I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_*n*), "I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_n)."

Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_*n*), "Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_n)."

4.4.1 Pin/ball startup and reset states

This table provides startup state and reset state information for device pins/balls.

Table 4-1. Pin/ball startup and reset states

Pin	Startup state[1]	State during reset	State after reset
I/O pins[2]	Weak pullup	Input, weak pullup	Input, weak pullup
Analog Inputs[3]	High-impedance	High-impedance	High-impedance

Table continues on the next page...

Table 4-1. Pin/ball startup and reset states (continued)

Pin	Startup state[1]	State during reset	State after reset
F0 (ERROR0, FI0)[4]	High-impedance	Input, high-impedance	Input, high-impedance
ESR0	Output	Output, open-drain	Input, weak pullup
ESR1	Weak pullup	Input, weak pullup	Input, weak pullup
PORST	Weak pulldown	Input, weak pulldown	Input, weak pulldown
TESTMODE[5]	Weak pulldown	Input, weak pulldown	Input, weak pulldown
JCOMP (TRST)	High-impedance	Input, weak pulldown	Input, weak pulldown
TDI	Weak pullup	Input, weak pullup	Input, weak pullup
TDO	Weak pullup	Weak pullup	High-impedance
TMS	High-impedance	Input, weak pullup	Input, weak pullup
TCK	High-impedance	Input, weak pulldown	Input, weak pulldown
XTAL/EXTAL[6]	High-impedance	High-impedance	High-impedance

1. Startup state is exited when the core and high-voltage supplies reach minimum levels as defined in the Power Management chapter.
2. Pins on the device that are not supply or reference are not explicitly defined in this table.
3. Pure analog inputs, on digital output function.
4. F0 is also referred to as ERROR0 and FI0.
5. The MPC5777M MCU does not implement a system configuration pin—TESTMODE is the only pin that affects device configuration. It is latched on the PORST rising edge and internally pulled-down.
6. Function of XTAL/EXTAL pins defined by the enable bit in the flash user test row. If enabled, the pins are analog inputs to the oscillator. If disabled, the XTAL/EXTAL pins default to TBD.

4.4.2 Power supply and reference voltage pins/balls

Power supply and reference pins/balls for the MPC5777M are listed in this section.

The following table contains information on power supply and reference pin functions for the devices.

NOTE

All ground supplies must be tied to ground. They can NOT float.

Table 4-2. Power supply and reference pins

Supply			BGA ball			
Symbol	**Type**	**Description**	**416PD**	**416ED**	**512PD**	**512ED**
V_{SS_HV}	Ground	High voltage ground	A26, B25, C24, D23, D15, D8, J4, L23, R23, T4, W23, AC23, AC19		B2, B29, B30, F6, F25, G7, G24, H29, H30, J9, J22, K10, K21,V29, AA21, AB22, AD24, AE25, AJ10, AJ29,AK10	

Table continues on the next page...

Table 4-2. Power supply and reference pins (continued)

Supply			BGA ball			
Symbol	**Type**	**Description**	**416PD**	**416ED**	**512PD**	**512ED**
V_{SS_LV}	Ground	Low voltage ground	K10, K11, K12, K15, K16, K17, L10, L11, L12, L13, L14, L15, L16, L17, M10, M11, M12, M13, M14, M15, M16, M17, N10, N11, N12, N13, N14, N15,N 16, N17, P11, P12, P13, P14, P15, P16, R 11, R12, R13, R14, R15, R16, T10, T11, T12, T13, T14, T15, T16, T17		M14, M17, N14, N15, N16, N17, P12, P13, P15, P16, P18, P19, R13, R14, R15, R16, R17, R18, T13,T14, T15, T16, T17, T18, U12, U13, U 15, U16, U18, U19, V14, V15 V16, V17, W14, W17	
$V_{SS\_HV\_ADV}$	Ground	Ground supply for ADC	—		—	
V_{DD_LV}	Power	Low voltage power supply for production device (PLL is also powered by this pin.)	B26, C25, D9, D24, E23, H4, P23, V23, AB23, AC20		M18, N19, V12, V19, W13, W18	
$V_{DD\_LV\_BD}$	Power	Low voltage power supply for buddy die	R1, R4		M13, N12	
$V_{DD\_HV\_PMC}$	Power	High voltage power supply for internal power management unit	D14		—	
$V_{DD\_HV\_IO\_MAIN}$	Power	High voltage power supply for I/O	A25, B24, C23, D22, K4, AC16, AD16, AE16, AF16		A2, A29, B3, B28, F7, F24, G8, G23, AC24, AD25, AH29, AJ30	
$V_{DD\_HV\_IO\_BD}$	Power	High voltage power supply for buddy die I/O	—	P17	—	R19
$V_{DD\_HV\_OSC}$	Power	Oscillator pin supply	E26		V25	
$V_{SS\_HV\_OSC}$	Ground	Oscillator ground supply	F25		T25	
$V_{DD\_HV\_JTAG}$	Power	JTAG pin supply	E26		V25	
$V_{DD\_HV\_IO\_FLEX}$	Power	FlexRay/Ethernet 3.3 V I/O supply	D7		J10	
$V_{DD\_HV\_IO\_FLEXE}$	Power	FLexRay/Ethernet/EBI I/O Segment Voltage Supply	AC18, AC22		AJ11, AK11, AK20, AK29	
$V_{DD\_HV\_IO\_EBI}$	Power	EBI Address/Control I/O Segment Voltage Supply	M23,T23,Y23		J29, J30, V30, AH30	
$V_{DD\_HV\_FLA}$	Power	Decoupling supply pin for flash	A18, B18		J21, K20	
$V_{DD\_HV\_ADV}$	Power	High voltage supply for ADC	—		—	
$V_{SS\_HV\_ADV\_S}$	Reference	Ground reference of ADC SAR	AF9		AF9, AJ8	
$V_{DD\_HV\_ADV\_S}$	Reference	Voltage reference of ADC SAR	AE9		AE10, AJ9	
$V_{SS\_HV\_ADV\_D}$	Reference	Ground reference of ADC σ/δ	AF5		AK8	
$V_{DD\_HV\_ADV\_D}$	Reference	Voltage reference of ADC σ/δ	AE5		AK9	
V_{DDSTBY}	Power	Standby RAM Supply Input	AD9		AA16	

4.4.3 System pins/balls

The following table contains information on system pin functions for the devices.

Table 4-3. System pins

Symbol	Description	Direction	BGA ball			
			416PD	416ED	512PD	512ED
PORST	Power on reset with Schmitt trigger characteristics and noise filter. PORST is active low	Bidirectional	B22		M22	
ESR0	External functional reset with Schmitt trigger characteristics and noise filter. ESR0 is active low	Bidirectional	A23		L21	
TESTMODE	Pin for testing purpose only TESTMODE pull-down is implemented to prevent the device from entering TESTMODE. It is recommended to connect the TESTMODE pin to VSS_HV_IO on the board. The value of the TESTMODE pin is latched at the negation of reset and has no affect afterward. **Note:** The device will not exit reset with the TESTMODE pin asserted during power-up.	Input only	B23		N24	
XTAL	Analog output of the oscillator amplifier circuit needs to be grounded if oscillator is used in bypass mode.	Output	G25		U24	
EXTAL	Analog input of the oscillator amplifier circuit when oscillator is not in bypass mode. Analog input for the clock generator when oscillator is in bypass mode.	Input	G26		U25	

4.4.4 LVDS pins/balls

The following table contains information on LVDS pin functions for the devices.

Table 4-4. LVDSM pin descriptions

Functional block	Port pin	Signal	Signal description	Direction	BGA ball (416 PD, 416 ED)	BGA ball (512 PD, 512 ED)
SIPI / LFAST[1]	PD[7]	SIPI_RXP	Interprocessor Bus LFAST, LVDS Receive Positive Terminal	I	G23	P24
	PF[13]	SIPI_RXN	Interprocessor Bus LFAST, LVDS Receive Negative Terminal	I	H23	R24
	PA[14]	SIPI_TXP	Interprocessor Bus LFAST, LVDS Transmit Positive Terminal	O	C26	T25

Table continues on the next page...

Table 4-4. LVDSM pin descriptions (continued)

Functional block	Port pin	Signal	Signal description	Direction	BGA ball (416 PD, 416 ED)	BGA ball (512 PD, 512 ED)
	PD[6]	SIPI_TXN	Interprocessor Bus LFAST, LVDS Transmit Negative Terminal	O	D26	R25
High-Speed Debug (HSD) / LFAST[1] [2]	PA[7]	DEBUG_TXP	Debug LFAST, LVDS Transmit Positive Terminal	O	F24	R21
	PA[8]	DEBUG_TXN	Debug LFAST, LVDS Transmit Negative Terminal	O	E25	N22
	PA[9]	DEBUG_RXP	Debug LFAST, LVDS Receive Positive Terminal	I	D25	N21
	PA[5]	DEBUG_RXN	Debug LFAST, LVDS Receive Negative Terminal	I	F23	T24
DSPI 4 Microsecond Bus	PD[2]	SCK_P	DSPI 4 Microsecond Bus Serial Clock, LVDS Positive Terminal	O	C18	F17
	PD[3]	SCK_N	DSPI 4 Microsecond Bus Serial Clock, LVDS Negative Terminal	O	C17	G18
	PD[0]	SOUT_P	DSPI 4 Microsecond Bus Serial Data, LVDS Positive Terminal	O	C16	F16
	PD[1]	SOUT_N	DSPI 4 Microsecond Bus Serial Data, LVDS Negative Terminal	O	D17	G16
DSPI 5 Microsecond Bus	PF[10]	SCK_P	DSPI 5 Microsecond Bus Serial Clock, LVDS Positive Terminal	O	J24	W24
	PF[9]	SCK_N	DSPI 5 Microsecond Bus Serial Clock, LVDS Negative Terminal	O	K23	W25
	PF[12]	SOUT_P	DSPI 5 Microsecond Bus Serial Data, LVDS Positive Terminal	O	J26	Y24
	PF[11]	SOUT_N	DSPI 5 Microsecond Bus Serial Data, LVDS Negative Terminal	O	J25	Y25
DSPI 6 Microsecond Bus	PQ[9]	SCK_P	DSPI 6Microsecond Bus Serial Clock, LVDS Positive Terminal	O	A17	A16
	PQ[8]	SCK_N	DSPI 6 Microsecond Bus Serial Clock, LVDS Negative Terminal	O	B17	B16

Table continues on the next page...

Table 4-4. LVDSM pin descriptions (continued)

Functional block	Port pin	Signal	Signal description	Direction	BGA ball (416 PD, 416 ED)	BGA ball (512 PD, 512 ED)
	PQ[11]	SOUT_P	DSPI 6 Microsecond Bus Serial Data, LVDS Positive Terminal	O	B16	A15
	PQ[10]	SOUT_N	DSPI 6 Microsecond Bus Serial Data, LVDS Negative Terminal	O	A16	B15
Differential DSPI 2	PD[2]	SCK_P	Differential DSPI 2 Clock, LVDS Positive Terminal	O	C18	F17
	PD[3]	SCK_N	Differential DSPI 2 Clock, LVDS Negative Terminal	O	C17	G17
	PD[0]	SOUT_P	Differential DSPI 2 Serial Output, LVDS Positive Terminal	O	C16	F16
	PD[1]	SOUT_N	Differential DSPI 2 Serial Output, LVDS Negative Terminal	O	D17	G16
	PD[7]	SIN_P	Differential DSPI 2 Serial Input, LVDS Positive Terminal	I	D17	P24
	PF[13]	SIN_N	Differential DSPI 2 Serial Input, LVDS Negative Terminal	I	H23	R24
Differential DSPI 5	PI[15]	SCK_P	Differential DSPI 5 Clock, LVDS Positive Terminal	O	G26	P22
	PI[14]	SCK_N	Differential DSPI 5 Clock, LVDS Negative Terminal	O	J23	R22
	PF[12]	SOUT_P	Differential DSPI 5 Serial Output, LVDS Positive Terminal	O	J26	Y24
	PF[11]	SOUT_N	Differential DSPI 5 Serial Output, LVDS Negative Terminal	O	J25	Y25
	PD[7]	SIN_P	Differential DSPI 5 Serial Input, LVDS Positive Terminal	I	G23	P24
	PF[13]	SIN_N	Differential DSPI 5 Serial Input, LVDS Negative Terminal	I	H23	R24

1. DRCLK and TCK/DRCLK usage for SIPI LFAST and Debug LFAST are described in the MPC5777M Microcontroller Reference Manual SIPI LFAST and Debug LFAST chapters.
2. Pads use special enable signal form DCI block: DCI driven enable for Debug LFAST pads is transparent to user.

Table 4-5. Aurora pin descriptions

Functional Block	PAD	Signal	Signal Description	Direction	BGA			
					416PD	416ED	512PD	512ED
Nexus Aurora High Speed Trace	—	TX0P	Nexus Aurora High Speed Trace Lane 0, LVDS Positive Terminal	O	—	U15	—	AB19
	—	TX0N	Nexus Aurora High Speed Trace Lane 0, LVDS Negative Terminal	O	—	U14	—	AB18
	—	TX1P	Nexus Aurora High Speed Trace Lane 1, LVDS Positive Terminal	O	—	U13	—	AB17
	—	TX1N	Nexus Aurora High Speed Trace Lane 1, LVDS Negative Terminal	O	—	U12	—	AB16
	—	TX2P	Nexus Aurora High Speed Trace Lane 2, LVDS Positive Terminal	O	—	U11	—	W16
	—	TX2N	Nexus Aurora High Speed Trace Lane 2, LVDS Negative Terminal	O	—	U10	—	W15
	—	TX3P	Nexus Aurora High Speed Trace Lane 3, LVDS Positive Terminal	O	—	P10	—	R12
	—	TX3N	Nexus Aurora High Speed Trace Lane 3, LVDS Negative Terminal	O	—	R10	—	T12
	—	CLKP (BD-AGBTCLKP)	Nexus Aurora High Speed Trace Clock, LVDS Positive Terminal	I	—	U17	—	AB21
	—	CLKN (BD-AGBTCLKN)	Nexus Aurora High Speed Trace Clock, LVDS Negative Terminal	I	—	U16	—	AB20

4.4.5 Generic pins/balls

The I/O Signal Description Table contains information on generic pins/balls. See the MPC5777M I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Multi-function pins are programmable via their respective Multiplexed Signal Configuration Register (MSCR) Source Signal Select (SSS) values. The SSS value selects which source signal is connected to the associated destination. For additional information, refer to SIUL2 chapter.

The I/O Signal Description Table provides the port pin, SIUL MSCR number, MSCR SSS, function, module, signal description, direction, pad type, and package pin/ball numbers for each I/O pin on the device.

Chapter 5
Memory Map

5.1 Overview memory maps

Table 5-1 shows the device memory map for the MPC5777M.

All addresses on the MPC5777M, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each region or module name. All memory not listed in this table is reserved, and access to those locations may produce undesirable results.

Table 5-1. Overview memory map

Start address	End address	Allocated size	Used size	Description
Flash (XBAR port 0—0)				
0x00000000	0x003FFFFF	—	—	Reserved
0x00400000	0x005FFFFF	2 MB		UTest NVM Block–no overlay
0x00600000	0x0067FFFF	512 KB	134 KB	HSM Code–no overlay
0x00680000	0x007FFFFF	1.5 MB	32 KB	HSM Data–no overlay
0x00800000	0x009FFFFF	2 MB	256 KB	Data Flash Blocks–no overlay
0x00A00000	0x00FFFFFF	6 MB	256 KB	Small and Medium Flash Blocks–no overlay
0x01000000	0x01FFFFFF	16 MB	3584 KB	Large Flash Blocks–no overlay
0x02000000	0x089FFFFF	8 MB	—	Reserved
0x02800000	0x02FFFFFF	8 MB	—	Reserved
0x03000000	0x037FFFFF	8 MB	—	Reserved
0x03800000	0x03FFFFFF	8 MB	—	Reserved
0x04000000	0x047FFFFF	8 MB	—	Reserved
0x04800000	0x04FFFFFF	8 MB	—	Reserved
0x05000000	0x057FFFFF	8 MB	—	Reserved
0x05800000	0x07FFFFFF	40 MB	—	Reserved
0x08000000	0x083FFFFF	4 MB	—	Reserved
0x08400000	0x089FFFFF	6 MB	—	Reserved
0x08A00000	0x08FFFFFF	6 MB	256 KB	Mirror Small and Medium Flash Blocks

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	Description
0x09000000	0x09FFFFFF	16 MB	3.5 MB	Mirror Large Flash Blocks
0x0A000000	0x07FFFFFF	8 MB	—	Reserved
0x0A800000	0x0AFFFFFF	8 MB	—	Reserved
0x0B000000	0x0B7FFFFF	8 MB	—	Reserved
0x0B800000	0x0BFFFFFF	8 MB		Reserved
0x0C000000	0x0C1FFFFF	2 MB	1 MB	Extended Overlay RAM[1]
0x0C200000	0x0C3FFFFF	2 MB	—	Reserved
0x0C400000	0x0C5FFFFF	2 MB	—	Reserved
0x0C600000	0x0C7FFFFF	2 MB	—	Reserved
0x0C800000	0x0CFFFFFF	8 MB	1 MB	Buddy Device Registers
0x0D000000	0x0D003FFF	16 KB	16 KB	Internal Overlay RAM
0x0D004000	0x0D00FFFF	48 KB	—	Reserved
0x0D010000	0x0FFFFFFF	49088 KB	—	Reserved
EBI (XBAR port 0—1)				
0x10000000	0x1FFFFFFF	256 MB		EBI - connection through P-flash controller (accesses use prefetch buffers and remap capability)
EBI (XBAR port 0—2)				
0x20000000	0x2FFFFFFF	256 MB		EBI - Direct Connection to the crossbar
Spare port (XBAR slave port 0—3)				
0x30000000	0x3FFFFFFF	256 MB	—	Reserved
System RAM (XBAR slave port 0—4)				
0x40000000	0x401FFFFF	2 MB		System RAM
0x40200000	0x407FFFFF	6 MB	—	Reserved
0x40800000	0x4FFFFFFF	248 MB	—	Reserved
Local memory (XBAR slave port 0—5)				
0x50000000	0x5FFFFFFF	256 MB		Processor Core local SRAM
Reserved port A (XBAR slave port 0—6)				
0x60000000	0x6FFFFFFF	256 MB	—	Reserved
Reserved port B (XBAR slave port 0—7)				
0x70000000	0x7FFFFFFF	256 MB	—	Reserved
Spare port (XBAR slave port 1—8)				
0x80000000	0x8FFFFFFF	256 MB	—	Reserved
Spare port (XBAR slave port 1—9)				
0x90000000	0x9FFFFFFF	256 MB	—	Reserved
HSM (Internal to HSM) (XBAR slave port 1—A)				
0xA0000000	0xA000FFFF	64 KB		Hardware Security Module (HSM)
0xA0010000	0xAFFFFFFF	262080 KB	—	Reserved
Spare port (XBAR slave port 1—B)				
0xB0000000	0xBFFFFFFF	262144 KB	—	Reserved
Reserved port C (XBAR slave port 1—C)				

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	Description
0xC0000000	0xCFFFFFFF	262144 KB	—	Reserved
Reserved port D (XBAR slave port 1—D)				
0xD0000000	0xDFFFFFFF	262144 KB	—	Reserved
Spare port (XBAR slave port 1—E)				
0xE0000000	0xEFFFFFFF	262144 KB	—	Reserved
Peripherals PBRIDGE_A, PBRIDGE_B (XBAR slave port 1—F)				
0xF0000000	0xFFFFFFFF	256 MB		Peripheral address space[-1]

1. Address range is for extended overlay RAM for buddy device.
2. See Table 2 for details.
3. See Table 2 for details.

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xF0000000	0xF7FFFFFF	131072 KB	—			Reserved
0xF8000000	0xF8003FFF	16 KB		B	PACR0	Peripheral Bridge B (PBRIDGE_B)
0xF8004000	0xFBE03FFF	—	—			Reserved
0xFBE04000	0xFBE07FFF	16 KB		B	OPACR126	SAR ADC 1 (SARADC_1)
0xFBE08000	0xFBE0BFFF	16 KB		B	OPACR125	SAR ADC 2 (SARADC_2)
0xFBE0C000	0xFBE0FFFF	16 KB		B	OPACR124	SAR ADC 3 (SARADC_3)
0xFBE10000	0xFBE13FFF	—	—			Reserved
0xFBE14000	0xFBE17FFF	16 KB		B	OPACR122	SAR ADC 5 (SARADC_5)
0xFBE18000	0xFBE1BFFF	16 KB		B	OPACR121	SAR ADC 6 (SARADC_6)
0xFBE1C000	0xFBE1FFFF	16 KB		B	OPACR120	SAR ADC 7 (SARADC_7)
0xFBE20000	0xFBE23FFF	16 KB		B	OPACR119	SAR ADC 8 (SARADC_8)
0xFBE24000	0xFBE27FFF	16 KB		B	OPACR118	SAR ADC 9 (SARADC_9)
0xFBE28000	0xFBE2BFFF	16 KB		B	OPACR117	SAR ADC 10 (SARADC_10)
0xFBE2C000	0xFBE3FFFF	—	—			Reserved
0xFBE40000	0xFBE43FFF	16 KB		B	OPACR111	Peripheral Sensor Interface 1 (PSI5_1)
0xFBE44000	0xFBE4FFFF	—	—			Reserved
0xFBE50000	0xFBE53FFF	16 KB		B	OPACR107	FlexRay 1 (FlexRay_1)
0xFBE54000	0xFBE5BFFF	—	—			Reserved
0xFBE5C000	0xFBE5FFFF	16 KB		B	OPACR104	SENT (SAE J2716) Receiver 1 (SRX_1)
0xFBE60000	0xFBE67FFF	—	—			Reserved
0xFBE68000	0xFBE6BFFF	16 KB		B	OPACR101	Inter IC Bus 1 (IIC_1)
0xFBE6C000	0xFBE6FFFF	—	—			Reserved
0xFBE70000	0xFBE73FFF	16 KB		B	OPACR99	Deserial Serial Peripheral Interface 2 (DSPI_2)
0xFBE74000	0xFBE77FFF	16 KB		B	OPACR98	Deserial Serial Peripheral Interface 3 (DSPI_3)

Table continues on the next page...

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map (continued)

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFBE78000	0xFBE7BFFF	16 KB		B	OPACR97	Deserial Serial Peripheral Interface 5 (DSPI_5)
0xFBE7C000	0xFBE8BFFF	—	—			Reserved
0xFBE8C000	0xFBE8FFFF	16 KB		B	OPACR92	LIN Controller 2 (LINFlexD_2)
0xFBE90000	0xFBEA7FFF	—	—			Reserved
0xFBEA8000	0xFBEABFFF	16 KB		B	OPACR85	LIN Controller 15 (LINFlexD_15)
0xFBEAC000	0xFBF0BFFF	—	—			Reserved
0xFBF0C000	0xFBF0FFFF	16 KB		B	OPACR60	Sigma Delta (SD) ADC 1 (SDADC_1)
0xFBF10000	0xFBF13FFF	16 KB		B	OPACR59	Sigma Delta (SD) ADC 3 (SDADC_3)
0xFBF14000	0xFBF17FFF	16 KB		B	OPACR58	Sigma Delta (SD) ADC 5 (SDADC_5)
0xFBF18000	0xFBF1BFFF	16 KB		B	OPACR57	Sigma Delta (SD) ADC 7 (SDADC_7)
0xFBF1C000	0xFBF1FFFF	16 KB		B	OPACR56	Sigma Delta (SD) ADC 9 (SDADC_9)
0xFBF20000	0xFBF57FFF	—	—			Reserved
0xFBF58000	0xFBF5BFFF	16 KB		B	OPACR41	Fault Collection and Control Unit (FCCU)
0xFBF5C000	0xFBF63FFF	—	—			Reserved
0xFBF64000	0xFBF67FFF	16 KB		B	OPACR38	Cyclic Redundancy Check 1 (CRC_1)
0xFBF68000	0xFBF73FFF	—	—			Reserved
0xFBF74000	0xFBF77FFF	16 KB		B	OPACR34	PSI5_S UART module (PSI5_S_0)
0xFBF78000	0xFBFB01FF	—	—			Reserved
0xFBFB0200	0xFBFB023F	64 Bytes		B	OPACR19	Clock Monitor Unit for Fast Crossbar (CMU_FXBAR)
0xFBFB0240	0xFBFB027F	64 Bytes		B	OPACR19	Clock Monitor Unit for Peripheral Shell Crossbar (CMU_SXBAR)
0xFBFB0280	0xFBFB02BF	64 Bytes		B	OPACR19	Clock Monitor Unit for Peripheral Bridge (CMU_PBRIDGE)
0xFBFB02C0	0xFBFB02FF	64 Bytes		B	OPACR19	Clock Monitor Unit for Peripherals (CMU_PER)
0xFBFB0300	0xFBFB03FF	64 Bytes		B	OPACR19	Clock Monitor Unit for Sigma Delta ADC (CMU_ADCSD)
0xFBFB0340	0xFBFB037F	64 Bytes		B	OPACR19	Clock Monitor Unit for SAR ADC (CMU_ADCSAR)
0xFBFB0380	0xFBFB03BF	64 Bytes		B	OPACR19	Clock Monitor Unit for SENT (CMU_SENT)
0xFBFB03C0	0xFBFB03FF	64 Bytes		B	OPACR19	Clock Monitor Unit for PSI5 (CMU_PSI5_F189)
0xFBFB0400	0xFBFB043F	64 Bytes		B	OPACR19	Clock Monitor Unit for PSI5 (CMU_PSI5_F125)
0xFBFB0440	0xFBFB047F	64 Bytes		B	OPACR19	Clock Monitor Unit for PSI5 (CMU_PSI5_1µS)
0xFBFB0480	0xFBFB04BF	64 Bytes		B	OPACR19	Clock Monitor Unit for Core (CMU_CORE)
0xFBFB04C0	0xFBFB04FF	64 Bytes		B	OPACR19	Clock Monitor Unit for CLKOUT (CMU_CLKOUT)

Table continues on the next page...

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map (continued)

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFBFB0500	0xFBFFFFFF	—	—			Reserved
0xFC000000	0xFC003FFF	16 KB		A	PACR0	Peripheral Bridge A (PBRIDGE_A)
0xFC004000	0xFC007FFF	16 KB		A	PACR1	Computational Shell Crossbar (XBAR_0)
0xFC008000	0xFC00BFFF	16 KB		A	PACR2	Peripheral Shell Crossbar (XBAR_1)
0xFC00C000	0xFC00FFFF	—	—			Reserved
0xFC010000	0xFC013FFF	16 KB		A	PACR4	System Memory Protection Unit 0 (SMPU_0)
0xFC014000	0xFC017FFF	16 KB		A	PACR5	System Memory Protection Unit 1 (SMPU_1)
0xFC018000	0xFC01BFFF	16 KB		A	PACR6	Cross Bar Integrity Checker 0 (XBIC_0)
0xFC01C000	0xFC01FFFF	16 KB		A	PACR7	Cross Bar Integrity Checker 1 (XBIC_1)
0xFC020000	0xFC023FFF	16 KB		A	PACR8	Platform RAM controller (PRAM_0)
0xFC024000	0xFC027FFF	—	—			Reserved
0xFC028000	0xFC02BFFF	16 KB		A	PACR10	Platform Configuration Module (PCM)
0xFC02C000	0xFC02FFFF	—	—			Reserved
0xFC030000	0xFC033FFF	16 KB		A	PACR12	PFLASH (Platform Flash controller)
0xFC034000	0xFC03BFFF	—	—			Reserved
0xFC03C000	0xFC03FFFF	16 KB		A	PACR15	Semaphore (SEMA4)
0xFC040000	0xFC043FFF	16 KB		A	PACR16	Interrupt Controller (INTC_0)
0xFC044000	0xFC04FFFF	—	—			Reserved
0xFC050000	0xFC053FFF	16 KB		A	PACR20	Software Watchdog Timer 0 (SWT_0)
0xFC054000	0xFC057FFF	16 KB		A	PACR21	Software Watchdog Timer 1 (SWT_1)
0xFC058000	0xFC05BFFF	16 KB		A	PACR22	Software Watchdog Timer 2 (SWT_2)
0xFC05C000	0xFC05FFFF	16 KB		A	PACR23	Software Watchdog Timer 3 (SWT_3)
0xFC06C000	0xFC067FFF	—	—			Reserved
0xFC068000	0xFC06BFFF	16 KB		A	PACR26	System Timer Module 0 (STM_0)
0xFC06C000	0xFC06FFFF	16 KB		A	PACR27	System Timer Module 1 (STM_1)
0xFC070000	0xFC073FFF	16 KB		A	PACR28	System Timer Module 2 (STM_2)
0xFC074000	0xFC07FFFF	—	—			Reserved
0xFC0A0000	0xFC0A3FFF	16 KB		A	PACR40	Direct Memory Access Controller 0 (DMA_0)
0xFC0A4000	0xFC0A7FFF	16 KB		A	PACR41	Direct Memory Access Controller 1 (DMA_1)
0xFC0A8000	0xFC0AFFFF	—	—			Reserved
0xFC0B0000	0xFC0B3FFF	16 KB			PACR44	Fast Ethernet Controller 0 (FEC_0)
0xFC0B4000	0xFC0E3FFF	—	Reserved			
0xFC0E4000	0xFC0E7FFF	16 KB		A	PACR57	Tamper Detect (Tamper_DETECT)
0xFC0E8000	0xFFCFFFFF	—	—			Reserved
0xFFD00000	0xFFDBFFFF	768 KB		A	OPACR144–OPACR191	GTM Integration (GTMINT) Module
0xFFDC0000	0xFFDFFFFF	—	—			Reserved

Table continues on the next page...

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map (continued)

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFE00000	0xFFE03FFF	16 KB		A	OPACR127	SARADC_0
0xFFE04000	0xFFE0FFFF	—	—			Reserved
0xFFE10000	0xFFE13FFF	16 KB		A	OPACR123	SAR ADC 4 (SARADC_4)
0xFFE14000	0xFFE3BFFF	—	—			Reserved
0xFFE3C000	0xFFE3FFFF	16 KB		A	OPACR112	Successive Approximation Register Analog-to-Digital Converter B (SARADC_B)
0xFFE40000	0xFFE43FFF	16 KB		A	OPACR111	Peripheral Sensor Interface 0 (PSI5_0)
0xFFE44000	0xFFE4FFFF	—	—			Reserved
0xFFE50000	0xFFE53FFF	16 KB		A	OPACR107	FlexRay Communication Controller 0 (FLEXRAY_0)
0xFFE54000	0xFFE5BFFF	—	—			Reserved
0xFFE5C000	0xFFE5FFFF	16 KB		A	OPACR104	SENT (SAE J2716) Receiver 0 (SRX_0)
0xFFE60000	0xFFE67FFF	—	—			Reserved
0xFFE68000	0xFFE6BFFF	16 KB		A	OPACR101	Inter IC Bus 0 (IIC_0)
0xFFE6C000	0xFFE6FFFF	—	—			Reserved
0xFFE70000	0xFFE73FFF	16 KB		A	OPACR99	Deserial Serial Peripheral Interface 0 (DSPI_0)
0xFFE74000	0xFFE77FFF	16 KB		A	OPACR98	Deserial Serial Peripheral Interface 1 (DSPI_1)
0xFFE78000	0xFFE7BFFF	16 KB		A	OPACR97	Deserial Serial Peripheral Interface 4 (DSPI_4)
0xFFE7C000	0xFFE7FFFF	16 KB		A	OPACR96	Deserial Serial Peripheral Interface 6 (DSPI_6)
0xFFE80000	0xFFE87FFF	—	—			Reserved
0xFFE88000	0xFFE8BFFF	16 KB		A	OPACR93	Deserial Serial Peripheral Interface 12 (DSPI_12)
0xFFE8C000	0xFFE8FFFF	16 KB		A	OPACR92	LIN Controller 0 (LINFlexD_0)
0xFFE90000	0xFFE93FFF	16 KB		A	OPACR91	LIN Controller 1 (LINFlexD_1)
0xFFE94000	0xFFEA7FFF	—	—			Reserved
0xFFEA8000	0xFFEABFFF	16 KB		A	OPACR85	LIN Controller 14 (LINFlexD_14)
0xFFEAC000	0xFFEAFFFF	16 KB		A	OPACR84	LIN Controller 16 (LINFlexD_16)
0xFFEB0000	0xFFED3FFF	—	—			Reserved
0xFFED4000	0xFFED7FFF	16 KB	20 KB	A	OPACR74	Shared CAN Message RAM
0xFFED8000	0xFFEDBFFF	16 KB		A	OPACR73	Shared CAN Message RAM (extended)
0xFFEDC000	0xFFEDFFFF	16 KB		A	OPACR72	Time Triggered Controller Area Network 0 (TTCAN_0)
0xFFEE0000	0xFFEE3FFF	—	—			Reserved
0xFFEE4000	0xFFEE7FFF	16 KB		A	OPACR70	CAN Subsystem: Controller Area Network 1 (MCAN_1)
0xFFEE8000	0xFFEEBFFF	16 KB		A	OPACR69	CAN Subsystem: Controller Area Network 2 (MCAN_2)

Table continues on the next page...

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map (continued)

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFEEC000	0xFFEEFFFF	16 KB		A	OPACR68	CAN Subsystem: Controller Area Network 3 (MCAN_3)
0xFFEF0000	0xFFEF3FFF	16 KB		A	OPACR67	CAN Subsystem: Controller Area Network 4 (MCAN_4)
0xFFEF4000	0xFFF0BFFF	—	—			Reserved
0xFFF0C000	0xFFF0FFFF	16 KB		A	OPACR60	Sigma Delta (SD) ADC 0 (SDADC_0)
0xFFF10000	0xFFF13FFF	16 KB		A	OPACR59	Sigma Delta (SD) ADC 2 (SDADC_2)
0xFFF14000	0xFFF17FFF	16 KB		A	OPACR58	Sigma Delta (SD) ADC 4 (SDADC_4)
0xFFF18000	0xFFF1BFFF	16 KB		A	OPACR57	Sigma Delta (SD) ADC 6 (SDADC_6)
0xFFF1C000	0xFFF1FFFF	16 KB		A	OPACR56	Sigma Delta (SD) ADC 8 (SDADC_8)
0xFFF20000	0xFFF2FFFF	—	—			Reserved
0xFFF30000	0xFFF33FFF	16 KB		A	OPACR51	Hardware Security Module (HSM) Interface[1]
0xFFF34000	0xFFF37FFF	—	—			Reserved
0xFFF38000	0xFFF3BFFF	16 KB		A	OPACR49	Development Tools Semaphore (DTS)
0xFFF3C000	0xFFF3FFFF	16 KB		A	OPACR48	JTAG Data Communication (JDC) Module
0xFFF40000	0xFFF43FFF	—	—			Reserved
0xFFF44000	0xFFF47FFF	16 KB		A	OPACR46	Self Test Control Unit (STCU2)
0xFFF48000	0xFFF4BFFF	16 KB		A	OPACR45	JTAG Master (JTAGM)
0xFFF4C000	0xFFF4FFFF	—	—			Reserved
0xFFF50000	0xFFF53FFF	16 KB		A	OPACR43	Memory Error Management Unit (MEMU)
0xFFF54000	0xFFF57FFF	16 KB		A	OPACR42	Indirect Memory Access Module (IMA)
0xFFF58000	0xFFF63FFF	—	—			Reserved
0xFFF64000	0xFFF67FFF	16 KB		A	OPACR38	Cyclic Redundancy Check 0 (CRC_0)
0xFFF68000	0xFFF6BFFF	—	—			Reserved
0xFFF6C000	0xFFF6C1FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 1 (DMAMUX_0)
0xFFF6C200	0xFFF6C3FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 1 (DMAMUX_1)
0xFFF6C400	0xFFF6C5FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer2 (DMAMUX_2)
0xFFF6C600	0xFFF6C7FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 3 (DMAMUX_3)
0xFFF6C800	0xFFF6C9FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 4 (DMAMUX_4)
0xFFF6CA00	0xFFF6CBFF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 5 (DMAMUX_5)
0xFFF6CC00	0xFFF6CDFF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 6 (DMAMUX_6)

Table continues on the next page...

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map (continued)

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFF6CE00	0xFFF6CFFF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 7 (DMAMUX_7)
0xFFF6D000	0xFFF6D1FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 8 (DMAMUX_8)
0xFFF6D200	0xFFF6D3FF	512 Byte		A	OPACR36	Direct Memory Access Controller Hardware Multiplexer 9 (DMAMUX_9)
0xFFF6D400	0xFFF73FFF	—	—			Reserved
0xFFF74000	0xFFF77FFF			A	OPACR34	Analog Test Module (ATX)
0xFFF78000	0xFFF7FFFF	—	—			Reserved
0xFFF80000	0xFFF83FFF	16 KB		A	OPACR31	Periodic Interval Timer 1 (PIT_1)
0xFFF84000	0xFFF87FFF	16 KB		A	OPACR30	Periodic Interval Timer 0 (PIT_0)
0xFFF88000	0xFFF97FFF	—	—			Reserved
0xFFF98000	0xFFF9BFFF	16 KB		A	OPACR25	Wake-Up Unit (WKPU)
0xFFF9C000	0xFFF9FFFF	—	—			Reserved
0xFFFA0000	0xFFFA03FF	1 KB		A	OPACR23	Power Control Unit (MC_PCU)
0xFFFA0400	0xFFFA3FFF	15 KB		A	OPACR23	Power Management Controller (PMCDIG)
0xFFFA4000	0xFFFA7FFF	—	—			Reserved
0xFFFA8000	0xFFFABFFF	16 KB		A	OPACR21	Reset Generation Module (MC_RGM)
0xFFFAC000	0xFFFAFFFF	—	—			Reserved
0xFFFB0000	0xFFFB003F	64 Byte		A	OPACR19	RCOSC (Internal RC Oscillator)
0xFFFB0040	0xFFFB007F	—	—			Reserved
0xFFFB0080	0xFFFB00BF	64 Byte		A	OPACR19	XOSC (Externally driven oscillator)
0xFFFB00C0	0xFFFB00FF	—	—			Reserved
0xFFFB0100	0xFFFB013F	64 Byte		A	OPACR19	FM Dual PLL (PLL)
0xFFFB0140	0xFFFB01FF	—	—			Reserved
0xFFFB0200	0xFFFB023F	64 Byte		A	OPACR19	Clock Monitor Unit for the I/O Processor (CMU_PLL)
0xFFFB0240	0xFFFB06FF	—	—			Reserved
0xFFFB0000	0xFFFB3FFF	14 KB		A	OPACR19	Clock Generation Module (MC_CGM)
0xFFFB4000	0xFFFB7FFF	—	—			Reserved
0xFFFB8000	0xFFFBBFFF	16 KB		A	OPACR17	Mode Entry Module (MC_ME)
0xFFFBC000	0xFFFBFFFF	—	—			Reserved
0xFFFC0000	0xFFFC3FFF	16 KB		A	OPACR15	System Integration Unit Lite 2 (SIUL2)
0xFFFC4000	0xFFFCFFFF	—	—			Reserved
0xFFFD0000	0xFFFD3FFF	16 KB		A	OPACR11	Serial Interprocessor Interface 0 (SIPI_0)
0xFFFD4000	0xFFFD7FFF	16 KB		A	OPACR10	Serial Interprocessor Interface 1 (SIPI_1) (Debug)

Table continues on the next page...

Table 5-2. Peripheral (PBRIDGE_A, PBRIDGE_B) memory map (continued)

Start address	End address	Allocated size	Used size	PBRIDGE	PBRIDGE Access Control Register	Description
0xFFFD8000	0xFFFDBFFF	16 KB		A	OPACR9	LFAST 0 (LFAST_0) (interprocessor communication)
0xFFFDC000	0xFFFDFFFF	16 KB		A	OPACR8	LFAST 1 (LFAST_1) (Debug)
0xFFFE0000	0xFFFE3FFF	16 KB		A	OPACR7	Flash main control registers
0XFFFE4000	0xFFFE7FFF	16 KB		A	OPACR6	Flash alternate program interface control registers
0xFFFE8000	0xFFFEFFFF	—	—			Reserved
0xFFFF0000	0xFFFF3FFF	16 KB		A	OPACR3	External Bus Interface (EBI)
0xFFFF4000	0xFFFF7FFF	16 KB		A	OPACR2	Password and Device Security Module (PASS)
0xFFFF8000	0xFFFFBFFF	16 KB		A	OPACR1	System Status and Configuration Module (SSCM)
0xFFFFC000	0xFFFFFFFF	16 KB		A	OPACR0	Boot Assist ROM (BAR)

1. This is the address space for the Host/HSM interface registers. See the MPC5777M Microcontroller Security Reference Manual for details.

5.2 Flash memory maps

Table 5-3 is the detailed flash memory map for the .

Table 5-3. Flash memory and overlay RAM map

Start address	End address	Allocated size [KB]	Complete flash block structure	RWW partition	Block size
Reserved—no overlay					
0x00000000	0x003FFFFF	—	Reserved		
UTest NVM block—no overlay					
0x00400000	0x00403FFF	16 KB	UTest NVM Block Space 16 KB	0	16 KB
0x00404000	0x00407FFF	16 KB	BAF (block0)	0	16 KB
0x00408000	0x005FFFFF	—	Reserved		
HSM Code—no overlay					
0x00600000	0x0060BFFF	48 KB	Reserved		
0x0060C000	0x0060FFFF	16 KB	16 KB HSM code (block5)	1	16 KB
0x00610000	0x0061FFFF	64 KB	64 KB HSM code (block2)	1	64 KB
0x00620000	0x0062FFFF	64 KB	64 KB HSM code (block3)	1	64 KB
0x00630000	0x0067FFFF	—	Reserved		
HSM Data— no overlay					
0x00680000	0x00683FFF	16 KB	16 KB HSM data block0	2	16 KB

Table continues on the next page...

Table 5-3. Flash memory and overlay RAM map (continued)

Start address	End address	Allocated size [KB]	Complete flash block structure	RWW partition	Block size
0x00684000	0x00687FFF	16 KB	16 KB HSM data block1	3	16 KB
0x00688000	0x007FFFFF	—	Reserved		
Data Flash–no overlay					
0x00800000	0x0080FFFF	64 KB	EEPROM Block0	4	64 KB
0x00810000	0x0081FFFF	64 KB	EEPROM Block1	4	64 KB
0x00820000	0x0082FFFF	64 KB	EEPROM Block2	4	64 KB
0x00830000	0x0083FFFF	64 KB	EEPROM Block3	4	64 KB
0x00840000	0x0084FFFF	64 KB	EEPROM Block4	5	64 KB
0x00850000	0x0085FFFF	64 KB	EEPROM Block5	5	64 KB
0x00860000	0x0086FFFF	64 KB	EEPROM Block6	5	64 KB
0x00870000	0x0087FFFF	64 KB	EEPROM Block7	5	64 KB
0x00880000	0x009FFFFF	—	Reserved		
Low & Mid Flash Blocks— no overlay					
0x00A00000	0x00FBFFFF	—	Reserved		
0x00FC0000	0x00FC3FFF	16 KB	16 KB Flash block1	0	16 KB
0x00FC4000	0x00FC7FFF	16 KB	16 KB Flash block2	0	16 KB
0x00FC8000	0x00FCBFFF	16 KB	16 KB Flash block3	1	16 KB
0x00FCC000	0x00FCFFFF	16 KB	16 KB Flash block4	1	16 KB
0x00FD0000	0x00FD7FFF	32 KB	32 KB Flash block0	0	32 KB
0x00FD8000	0x00FDFFFF	32 KB	32 KB Flash block1	1	32 KB
0x00FE0000	0x00FEFFFF	64 KB	64 KB Low Flash block0	0	64 KB
0x00FF0000	0x00FFFFFF	64 KB	64 KB Low Flash block1	0	64 KB
Large Flash Blocks—no overlay					
0x01000000	0x0103FFFF	256 KB	256 KB Flash block0	6	256 KB
0x01040000	0x0107FFFF	256 KB	256 KB Flash block1	6	256 KB
0x01080000	0x010BFFFF	256 KB	256 KB Flash block2	6	256 KB
0x010C0000	0x010FFFFF	256 KB	256 KB Flash block3	6	256 KB
0x01100000	0x0113FFFF	256 KB	256 KB Flash block4	6	256 KB
0x01140000	0x0117FFFF	256 KB	256 KB Flash block5	6	256 KB
0x01180000	0x011BFFFF	256 KB	256 KB Flash block6	6	256 KB
0x011C0000	0x011FFFFF	256 KB	256 KB Flash block7	6	256 KB
0x01200000	0x0123FFFF	256 KB	256 KB Flash block8	7	256 KB
0x01240000	0x0127FFFF	256 KB	256 KB Flash block9	7	256 KB
0x01280000	0x012BFFFF	256 KB	256 KB Flash block10	7	256 KB
0x012C0000	0x012FFFFF	256 KB	256 KB Flash block11	7	256 KB
0x01300000	0x0133FFFF	256 KB	256 KB Flash block12	7	256 KB
0x01340000	0x0137FFFF	256 KB	256 KB Flash block13	7	256 KB
0x01380000	0x013BFFFF	256 KB	256 KB Flash block14	7	256 KB
0x013C0000	0x013FFFFF	256 KB	256 KB Flash block15	7	256 KB

Table continues on the next page...

Table 5-3. Flash memory and overlay RAM map (continued)

Start address	End address	Allocated size [KB]	Complete flash block structure	RWW partition	Block size
0x01400000	0x0143FFFF	256 KB	256 KB Flash block16	8	256 KB
0x01440000	0x0147FFFF	256 KB	256 KB Flash block17	8	256 KB
0x01480000	0x014BFFFF	256 KB	256 KB Flash block18	8	256 KB
0x014C0000	0x014FFFFF	256 KB	256 KB Flash block19	8	256 KB
0x01500000	0x0153FFFF	256 KB	256 KB Flash block20	8	256 KB
0x01540000	0x0157FFFF	256 KB	256 KB Flash block21	8	256 KB
0x01580000	0x015BFFFF	256 KB	256 KB Flash block22	8	256 KB
0x015C0000	0x015FFFFF	256 KB	256 KB Flash block23	9	256 KB
0x01600000	0x0163FFFF	256 KB	256 KB Flash block24	9	256 KB
0x01640000	0x0167FFFF	256 KB	256 KB Flash block25	9	256 KB
0x01680000	0x016BFFFF	256 KB	256 KB Flash block26	9	256 KB
0x016C0000	0x016FFFFF	256 KB	256 KB Flash block27	9	256 KB
0x01700000	0x0173FFFF	256 KB	256 KB Flash block28	9	256 KB
0x01740000	0x0177FFFF	256 KB	256 KB Flash block29	9	256 KB
0x01780000	0x01FFFFFF	—	Reserved		
Reserved Flash—no overlay					
0x02000000	0x07FFFFFF	—	Reserved		
Reserved Flash—no overlay					
0x03800000	0x07FFFFFF	—	Reserved		
Mirror Reserved Flash					
0x08000000	0x089FFFFF	—	Reserved		
Low & Mid flash blocks—overlay enabled mapping **(Mirror of flash blocks starting at 0x00FC_0000 but with RAM overlay allowed)**					
0x08A00000	0x08FBFFFF	—	Reserved		
0x08FC0000	0x08FC3FFF	16 KB	16 KB Flash block1	0	16 KB
0x08FC4000	0x08FC7FFF	16 KB	16 KB Flash block2	0	16 KB
0x08FC8000	0x08FCBFFF	16 KB	16 KB Flash block3	1	16 KB
0x08FCC000	0x08FCFFFF	16 KB	16 KB Flash block4	1	16 KB
0x08FD0000	0x08FD7FFF	32 KB	32 KB Flash block0	0	32 KB
0x08FD8000	0x08FDFFFF	32 KB	32 KB Flash block1	1	32 KB
0x08FE0000	0x08FEFFFF	64 KB	64 KB Low Flash block0	0	64 KB
0x08FF0000	0x08FFFFFF	64 KB	64 KB Low Flash block1	0	64 KB
Large flash blocks—overlay enabled mapping					
0x09000000	0x0903FFFF	256 KB	256 KB Flash block0	6	256 KB
0x09040000	0x0907FFFF	256 KB	256 KB Flash block1	6	256 KB
0x09080000	0x090BFFFF	256 KB	256 KB Flash block2	6	256 KB
0x090C0000	0x090FFFFF	256 KB	256 KB Flash block3	6	256 KB
0x09100000	0x0913FFFF	256 KB	256 KB Flash block4	6	256 KB
0x09140000	0x0917FFFF	256 KB	256 KB Flash block5	6	256 KB

Table continues on the next page...

Table 5-3. Flash memory and overlay RAM map (continued)

Start address	End address	Allocated size [KB]	Complete flash block structure	RWW partition	Block size
0x09180000	0x091BFFFF	256 KB	256 KB Flash block6	6	256 KB
0x091C0000	0x091FFFFF	256 KB	256 KB Flash block7	7	256 KB
0x09200000	0x0923FFFF	256 KB	256 KB Flash block8	7	256 KB
0x09240000	0x0927FFFF	256 KB	256 KB Flash block9	7	256 KB
0x09280000	0x092BFFFF	256 KB	256 KB Flash block10	7	256 KB
0x092C0000	0x092FFFFF	256 KB	256 KB Flash block11	7	256 KB
0x09300000	0x0933FFFF	256 KB	256 KB Flash block12	7	256 KB
0x09340000	0x0937FFFF	256 KB	256 KB Flash block13	7	256 KB
0x09380000	0x093BFFFF	256 KB	256 KB Flash block14	7	256 KB
0x093C0000	0x093FFFFF	256 KB	256 KB Flash block15	7	256 KB
0x09400000	0x0943FFFF	256 KB	256 KB Flash block16	8	256 KB
0x09440000	0x0947FFFF	256 KB	256 KB Flash block17	8	256 KB
0x09480000	0x094BFFFF	256 KB	256 KB Flash block18	8	256 KB
0x094C0000	0x094FFFFF	256 KB	256 KB Flash block19	8	256 KB
0x09500000	0x0953FFFF	256 KB	256 KB Flash block20	8	256 KB
0x09540000	0x0957FFFF	256 KB	256 KB Flash block21	8	256 KB
0x09580000	0x095BFFFF	256 KB	256 KB Flash block22	8	256 KB
0x095C0000	0x095FFFFF	256 KB	256 KB Flash block23	9	256 KB
0x09600000	0x0963FFFF	256 KB	256 KB Flash block24	9	256 KB
0x09640000	0x0967FFFF	256 KB	256 KB Flash block25	9	256 KB
0x09680000	0x096BFFFF	256 KB	256 KB Flash block26	8	256 KB
0x096C0000	0x096FFFFF	256 KB	256 KB Flash block27	9	256 KB
0x09700000	0x0973FFFF	256 KB	256 KB Flash block28	9	256 KB
0x09740000	0x0977FFFF	256 KB	256 KB Flash block29	9	256 KB
0x09780000	0x09FFFFFF	—	Reserved		
Reserved flash—overlay enabled mapping					
0x0A000000	0x0A7FFFFF	—	Reserved		
Reserved flash—overlay enabled mapping					
0x0B000000	0x0BFFFFFF	—	Reserved		
Buddy Overlay RAM					
0x0C000000	0x0C1FFFFF	2 MB	Extended Overlay RAM	—	2 MB[1]
0x0C200000	0x0C7FFFFF	—	Reserved		
0x0C800000	0x0CFFFFFF	8 MB	Buddy Device Registers		
Internal Overlay RAM					
0x0D000000	0x0D003FFF	16 KB	Internal Overlay RAM		16 KB
0x0D004000	0x0FFFFFFF	—	Reserved		

1. Consists of eight 256 KB regions.

Table 5-4 is the memory map for the buddy device.

Table 5-4. Buddy device memory map

Start address	End address	Allocated size [KB]	Description	BD2M
Buddy Overlay RAM				
0x0C000000	0x0C0FFFFF	1024 KB	Extended Overlay RAM	1M
0x0C100000	0x0C1FFFFF	1024 KB	Reserved Buddy Overlay RAM	1M
0x0C200000	0x0C7FFFFF	—	Reserved	
Buddy registers				
0x0C800000	0x0C803FFF	16 KB	System Integration Unit Lite 2 (SIUL2)	
0x0C804000	0x0C807FFF	16 KB	LFAST	
0x0C808000	0x0C80BFFF	16 KB	JTAGM	
0x0C80C000	0x0C80FFFF	16 KB	DWPU	
0x0C810000	0x0C813FFF	16 KB	Buddy Device Crossbar	
0x0C814000	0x0CFFFFFF	—	Reserved	

5.3 System RAM memory maps

The following table details the system RAM memory map.

Table 5-5. RAM memory map

Start address	End address	Allocated size [KB]	Used size [KB]	Description
System RAM (XBAR Port 0—4)				
0x40000000	0x4000FFFF	64 KB	64 KB	Standby RAM
0x40010000	0x401FFFFF	2048 KB	340 KB	System RAM
0x40200000	0x4FFFFFFF	—	—	Reserved
Local Memory (XBAR Port 0—5)				
0x50000000	0x5000FFFF	64 KB	16 KB	I-MEM CPU0
0x50010000	0x507FFFFF	—	—	Reserved
0x50800000	0x5080FFFF	64 KB	64 KB	D-MEM CPU0
0x50810000	0x50FFFFFF	—	—	Reserved
0x51000000	0x5100FFFF	64 KB	16 KB	I-MEM CPU1
0x51010000	0x517FFFFF	—	—	Reserved
0x51800000	0x5180FFFF	64 KB	64 KB	D-MEM CPU1
0x51810000	0x51FFFFFF	8128 KB	—	Reserved
0x52000000	0x5200FFFF	64 KB	16 KB	I-MEM CPU2
0x52010000	0x527FFFFF	8128 KB	—	Reserved
0x52800000	0x5280FFFF	64 KB	64 KB	D-MEM CPU2
0x52810000	0x5FFFFFFF	8128 KB	—	Reserved

Table 5-6. UTest flash memory map

Start address	End address	Allocated size [bytes]	Description	Comments
UTest Flash				
0x00400000	0x00400001	2	Temperature Sensor Calibration A	Contains the fitting parameter TSCA extracted during factory-calibration to be used by software on temperature formula in order to calculate die temperature (see formula in the Temperature Sensor Chapter).
0x00400002	0x00400003	2	Temperature Sensor Calibration B	Contains the fitting parameter TSCB extracted during factory-calibration to be used by software on temperature formula in order to calculate die temperature (see formula Temperature Sensor Chapter).
0x00400004	0x0040000B	8	Reserved	—
0x0040000C	0x0040000F	4	Test Mode Disable Seal	Programmed by JDP when flash testing is complete
0x00400010	0x0040001F	16	Test Mode Disable Block Select Group A	Programmed by the customer to select which blocks cannot be tested in Failure Analysis
0x00400020	0x0040002F	16	Factory Erase diary Location	Programmed by the customer before one-time factory erase is allowed
0x00400030	0x0040003F	16	Test Mode Disable Block Select Group B	Secondary location that can be programmed by the customer to select which blocks CANNOT be tested in Failure Analysis
0x00400040	0x0040005F	32	Customer Single Bit Correction Area	Programmed by factory to include ECC/EDC errors to allow testing of ECC/EDC hardware
0x00400060	0x0040007F	32	Customer Double Bit Detection Area	256 bits — Reserved for Double Bit Detection (Double bit data programmed by Freescale).
0x00400080	0x0040009F	32	Customer EDC after ECC Area	256 bits — Reserved for EDC after ECC Detection (Data programmed by Freescale).
0x004000A0	0x004000BF	32	Unique Identifier (UID)	Programmed during factory test.The UID includes information on Wafer lot, X/Y-position on the wafer, manufacturing data, test results, company ID: 65 = Freescale
0x004000C0	0x004000C3	4	Soft DCF Record Start Address	Programmed by the customer to indicate a non-standard start address for DCF Records targeting 'Soft' DCF clients
0x004000C4	0x004000FF	60	Reserved	

Table continues on the next page...

Table 5-6. UTest flash memory map (continued)

Start address	End address	Allocated size [bytes]	Description	Comments
0x00400100	0x00400103	4	Test Mode Override Passcode	Protected from read operations by flash BIU from Production at OEM Life Cycle. Programmed by customer. Becomes unreadable (hidden) as soon as customer programs lifecycle stage to 'Production at OEM' or later
0x00400104	0x0040011F	28	Fuse Bypass Password	
0x00400120	0x0040013F	32	JTAG Password	
0x00400140	0x0040015F	32	PASS Password Group 0	
0x00400160	0x0040017F	32	PASS Password Group 1	
0x00400180	0x0040019F	32	PASS Password Group 2	
0x004001A0	0x004001BF	32	PASS Password Group 3	
0x004001C0	0x004001FF	64	Reserved	
0x00400200	0x0040020F	16	Lifecycle Slot 0 - JDP_PROD	"JDP Production" Lifecycle slot. Programmed by JDP during test
0x00400210	0x0040021F	16	Lifecycle Slot 1 - CUST_DEL	"Customer Delivery" Lifecycle slot. Programmed by JDP during test
0x00400220	0x0040022F	16	Lifecycle Slot 2 - OEM_PROD	"OEM Production" Lifecycle slot. Programmed by customer during ECO module production/test
0x00400230	0x0040023F	16	Lifecycle Slot 3 - IN_FIELD	"In Field" Lifecycle slot. Programmed by customer during ECO module production/test
0x00400240	0x0040024F	16	Lifecycle Slot 4 - FA	Failure Analysis Lifecycle slot. Programmed by the customer on returned modules
0x00400250	0x004002FF	176	Reserved	
0x00400300	0x00400307	8	DCF Start Record	DCF start Record. Programmed by JDP during Test. Contiguous list of DCF records starts at this address
0x00400308	0x00400347	64	DCF HSM 'ROM' key Records	Space for 8 DCF records used to load the HSM 'ROM' keys. **Note:** This region is only fully readable while Lifecycle = JDP_PROD. When Lifecycle has reached CUST_DEL, this region is read protected by the flash memory flash controller (PFLASH). Reads to this region return dummy DCF records, allowing software to parse the complete DCF record list, from start record to stop record
0x00400348	0x00400FFF	3256	DCF Records	Contiguous list of DCF Records starting at 0x00400348. Initial records programmed by factory. Subsequent records added to the end of the list by the customer
0x00401000	0x00403FFF	12288	Reserved for customer OTP data	Programmed by the customer

Table continues on the next page...

Table 5-6. UTest flash memory map (continued)

Start address	End address	Allocated size [bytes]	Description	Comments
0x00404000	0x00407FFF	16384	BAF code	Reserved for Boot Assist Flash (BAF). This is factory code that executes every reset to implement boot features. It is programmed at the factory and is OTP

Chapter 6
Functional Safety

6.1 Introduction

The MPC5777M offers a set of features to support applications that need to fulfill functional safety requirements as defined by automotive safety integrity level ASIL D of ISO 26262. This chapter gives an overview of the safety implementation for the MPC5777M, and discusses the operation of the safety mechanisms.

6.2 Safety overview

The MPC5777M has been designed for automotive applications executing safety-relevant tasks that require a fail-silent or fail-indicate MCU. The MPC5777M supports a Fault Tolerant Time Interval (FTTI) of at least 10 ms. Shorter intervals are possible depending on the actual combination of failure detection and signaling mechanisms as documented in the Safety Manual (SM), but the MPC5777M generally supports an FTTI of 10 ms.

6.3 Module categorization

The definition of the MPC5777M safety concept and its validation (particularly the computation of Probabilistic Metric for Random Hardware Failures (PMHF)) has been done by splitting system modules into four categories as shown in Table 6-1.

Table 6-1. PMHF module categories

Module category	Category abbreviation	Description
Vital system modules	ViMo	All modules that can directly influence the correct operation of the software execution (by influence of the correct function of CPU, RAM, flash, and the bus interconnect). Examples of ViMo are the core, the cache, the crossbar, and the interrupt controller (INTC), as well as clock or power generation.
Peripheral modules	PeMo	All microcontroller modules involved in I/O operation, specifically including the I/O bridge.
Non-safety modules	NoSaMo	Modules that belong in one of the above categories but that do not require hardware-level safety measures. Examples include the DMA, the performance core, and the I/O core. Only their non-interference with modules from the other categories is ensured by hardware. In case they are used in a safety-relevant way, application measures are necessary to ensure their safe operation.
Other support modules	SuMo	These are modules that, although safety-relevant and in principle potential members of the ViMo category, cannot lead to a violation of the safety goal on their own. Typically, these will only have multiple point faults, but not single point faults. Example SuMo's are the RCCU and Clock Monitoring Unit (CMU).

Table 6-2 shows module classifications for the MPC5777M.

Table 6-2. Module classification

Classification	Modules
ViMos	Master Safety Core (Z7), Master Safety Core D-Cache controller, Master Safety Core I-Cache controller, Master Safety Core D-MEM controller, Master Safety Core I-MEM controller, Safety Core D-Cache array, Safety Core I-Cache array, Safety Core Local D-Mem array, Safety Core Local I-Mem array, INTC, XBAR_0 (Computational - Fast), SMPU, SRAM controller, system RAM array, flash memory controller, flash memory array, Overlay RAM, STM_0 (System Timer), MC_RGM, MC_RSL,

Table continues on the next page...

Table 6-2. Module classification (continued)

Classification	Modules
	Dual PLL (Digital and Analog), XOSC, MC_CGM (ViMos portion), MC_ME (ViMos portion), DSMC (ViMos portion), PMC (Digital and Analog - Voltage Regulator), STCU, SSCM, IAHB Gaskets, PIT
PeMos	SARADC (Digital, Analog, AMUX5 and BIAS), SDADC (Digital, Analog and Bandgap), PSI5, PSI5_S, SENT, LINFlexD, DSPI, FlexRay, FlexRay DRAM array, FlexRay LRAM array, FEC, FEC FIFO array, FEC MIB array, SIUL2, XBAR_1 (Peripheral - Slow), BUSC (Concentrator - FEC, FlexRay and LFAST), AIPS MC_CGM (PeMos portion), MC_ME (PeMos portion),
SuMos	RCCU, Checker Safety Core (Z7), Checker Safety Core D-Cache controller, Checker Safety Core I-Cache controller, Checker Safety Core D-MEM controller, Checker Safety Core I-MEM controller, CRC units, Temperature Sensors,

Table continues on the next page...

Table 6-2. Module classification (continued)

Classification	Modules
	IRCOSC, CMU, PMC-HVD12, PMC-HVD33, PMC-LVD12, PMC-LVD33, LBIST, MBIST, IMA, REG_PROT, SWT_0, FCCU, MEMU CGL (Clock Gating Logic), MC_CGM (SuMos portion), MC_ME (SuMos portion), PTM, PMC (Standby RAM)
NoSaMos	Performance Core (Z7), Peripheral Core (Z4), Performance Core D-Cache controller, Performance Core I-Cache controller, Performance Core D-MEM controller, Performance Core I-MEM controller, Performance Core D-Cache array, Performance Core I-Cache array, Performance Core Local D-Mem array, Performance Core Local I-Mem array, Peripheral Core I-Cache controller, Peripheral Core D-MEM controller, Peripheral Core I-MEM controller, Peripheral Core I-Cache array, Peripheral Core Local D-Mem array, Peripheral Core Local I-Mem array, EBI (LFAST, SIPI), eDMA, GTM, HSM

Table 6-2. Module classification

Classification	Modules
	DMAMUX, Nexus, JTAG (JTAGC, JTAGM), BUSC (Concentrator - DMA and HSM), I2C, SEMA4, WKPU SPU PASS, DCI DTS, JDC, NAR, NAR RAM Array, STM_1 and STM_2, SWT_1, SWT_2, and SWT_3, DSMC (NoSaMos portion),

The MPC5777M implements no specific protection of NoSaMos against Common Cause Failures between them.

6.4 System implementation

6.4.1 General concept

In general, safety integrity is achieved in the following ways:

- For the Safety Core and its closely related periphery, safety is ensured by a delayed lockstep approach (see Dual-core lockstep).
- The safety of storage and of the data path to storage and periphery is ensured by End-to-End ECC (E2E ECC) with address encoding and selected additional measures for individual modules. For the periphery, end-to-end ends at the I/O bridges (see End-to-End protection on data path).
- Clock-and-power generation and distribution are supervised by dedicated monitors (see Clock and power monitoring).

- The safety of the periphery is ensured by application-level measures (such as connecting one sensor to different I/O modules, sensor validation by sensor fusion, and so on). Hardware supports this application-level redundancy by providing redundant I/O modules connected to different peripheral bridges (PBRIDGE) to maximize the independence between the monitored and monitoring resources (see I/O peripherals, and Communication controllers).
- MBIST and LBIST are provided to avoid the accumulation of latent faults in the functional logic as well as the safety integrity mechanisms (see Built-In Self-Tests (BIST)). Dedicated mechanisms are provided to check the availability of safety mechanisms and the functionality of each error reaction path (such as by fake fault injection). Table 6-3 and Table 6-4 detail the available LBIST partitions.
- The Fault Collection and Control Unit (see "Fault Collection and Control Unit (FCCU)" chapter for FCCU details) is responsible for collecting and reacting to failure notifications (see FCCU and failure monitoring). MEMU (see Memory Error Management Unit (MEMU)) is responsible for the collection and reporting (to the FCCU) of error events in system memories, and flash memory, as well as E2E ECC errors caused by the XBAR, or RAM or flash memory controllers. Error events include ECC corrections and detections as well as faults detected by memory BISTs.
- Common Cause Failures (CCF) are dealt with by a set of measures for both control and avoidance of CCFs spanning system-level approaches (such as temperature and nonfunctional signal monitoring) and back-end techniques (such as isolated silicon areas and routing constraints) (see Common Cause Failure measures).

6.4.1.1 End-to-End protection on data path

Connections between XBAR masters and slaves (clients) are denoted as data paths. Data corruption on all data paths between the Safety Core and any client is detected with at least 99% coverage via two main safety mechanisms: data from the replicated cores is encoded using Error Correcting Code (ECC), which is implemented with a single-error correction, double-error detection (SECDED) code with a Hamming distance of 4 and includes coverage of addressing information; control signals and address decoding are monitored to verify the data reaches all of the intended clients, from all possible connections to these clients and the intended operation is performed on the target address. The following figure shows a general view of ECC schema.

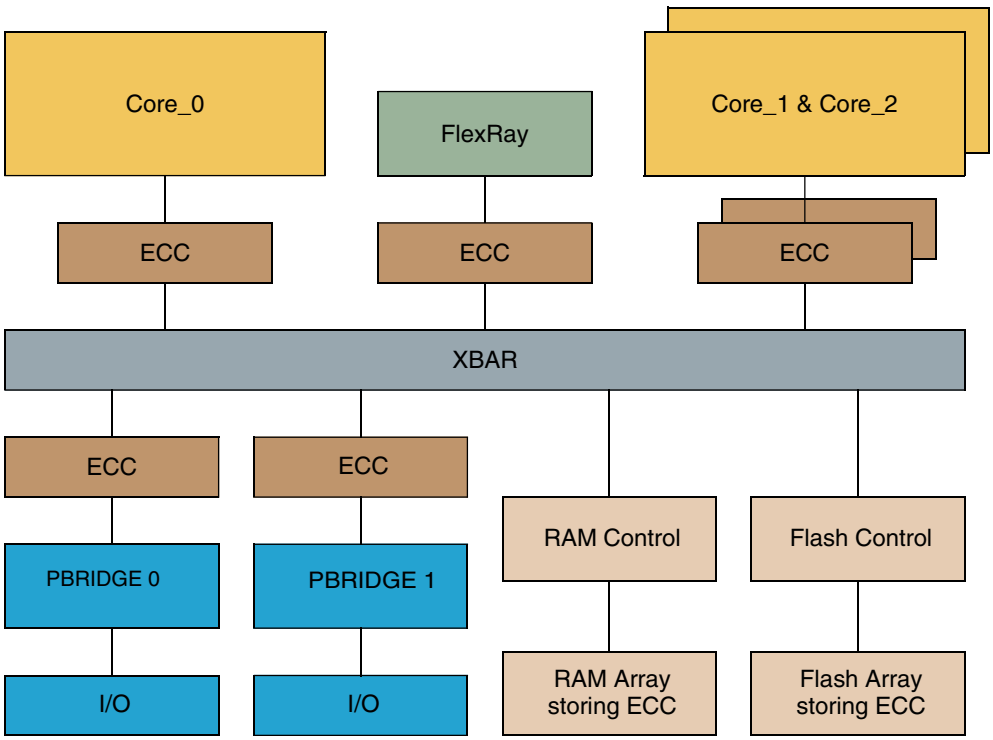


Figure 6-1. General view of E2E ECC

Note

Specific implementation for the MPC5777M vary depending on the special requirements of RAM and flash memory concerning ECC handling.

ECC bits are generated on writes by XBAR masters (including, but not limited to the Safety Core) and checked on reads. The ECC correction bits are stored alongside the data in flash memory and RAM so, in principle, no ECC logic is necessary at the memories themselves. For this reason the ECC schema is referred to as End-to-End ECC (E2E ECC) in the following sections. For XBAR slaves, apart from memories, new ECC logic is added as these clients cannot store or produce the ECC correction bits. This resolves the problem where ECC needs to be calculated in real time before entering or exiting the ECC-protected data path. This setup leaves the data path downstream of the ECC units (for example, starting at, and downstream of, the peripheral bridges) unprotected by ECC. To detect corruptions introduced on those unprotected data paths, the PBRIDGEs and peripherals can be used redundantly (see I/O peripherals), or other mechanisms may be used to validate peripheral operation.

The E2E ECC schema provides high detection capabilities against failures affecting the data content of the transaction. The inclusion of the target address in the computation of the redundancy bits (8 ECC bits) does allow the partial detection of addressing faults as well. To reach the desired integrity level, additional dedicated safety mechanisms are implemented in the data path particularly to:

- Improve the detection capability over addressing failures (no/multiple/wrong address selected), considering faults affecting address transmission (from master to client) as well as the decoding of the address;
- Provide coverage for control failures affecting, for example, the type (read vs. write) or size of a transaction.

Though safety mechanisms protecting the XBAR, the RAM controller, or the flash memory controller are different, they are all based on the feedback of address and control information from the target to the source of the transaction, which is responsible for checking for consistency with respect to the intended transaction. Depending on the portion of the data path covered by the specific safety mechanism, the source can be an XBAR master port rather than the XBAR interface of the RAM or Flash controllers; the target is respectively an XBAR slave port, the RAM array, or the Flash module (see "Crossbar Switch (XBAR)", "Flash memory controller", and "RAM controller (PRAMC)" chapters).

6.4.1.2 Clock and power monitoring

6.4.1.2.1 Clock

Clocks in the MPC5777M are supervised by Clock Monitor Units (CMUs). The CMUs are driven by the IRCOSC (16 MHz internal oscillator) for independent operation from the monitored clocks. CMUs flag errors associated with conditions due to loss of clock, clock out of a programmable bound (over and under frequency), and loss of reference. If a supervised clock leaves the specified range for the device, a signal is sent to the FCCU.

The clock tree for Core_0 and Core_0 – Checker (the two lockstep cores comprising the Safety Core) consists of separate clock branches that are not shared with any other module. One CMU (CMU_1) is connected to the Main Core clock path, downstream from the common part of the clock tree for the Main and Checker Cores. In this way any frequency failure affecting the Main Core clock or the common clock (SYS_CLK) is detected by the CMU, and any frequency failure affecting the Checker Core clock is detected by the Redundancy Control and Checker Units (RCCUs) (see MCU clocking details).

6.4.1.2.2 Power

There are two types of voltage supervisors on the MPC5777M, Low Voltage Detect (LVD) and High Voltage Detect (HVD) monitors. Safety-relevant voltages are supervised for voltages that are out of these ranges. Since safety relevant voltages have the potential to disable the failure indication mechanisms of the MCU (such as FCCU, pads, and so on) the indication of these errors can be used to cause a direct transition of the MCU into the safe state (reset assertion) (see "Power management" and "Power Control Unit (MC_PCU)" chapters for details on power monitoring).

6.4.1.3 I/O peripherals

To allow a safety application to make redundant use of all I/O peripherals, each peripheral has at least two instances (except for the GTM), and each instance is connected to a different PBRIDGE. This means, for example, that if DSPI is provided by the MCU, two DSPI modules (DSPI*n*, DSPI*m*) are included and connected externally through different pins. Internally, DSPI*n* would then be connected to PBRIDGE_A and DSPI*m* to PBRIDGE_B and be accessible via different addresses.

The arrangement of I/O peripherals onto two PBRIDGEs, as well as further CCF prevention measures, allow redundant use of peripherals while limiting possible causes of CCFs. Redundant usage includes usage of equivalent peripherals in a replicated way as well as usage of functionally different peripherals in, for example, feedback measurement loops. Comparison of redundant operation is the responsibility of the application software, not the safety hardware mechanism. The allocation of peripherals along the two PBRIDGEs are depicted in block diagrams found in the “Introduction” chapter of this reference manual.

6.4.1.4 Communication controllers

Communication controllers provide the ability to exchange information with external components and are therefore subject to the same safety criteria as I/O peripherals. Yet we assume that for high bandwidth communication controllers, additional software measures are employed that do not require redundant communication peripherals.

The following communication controllers do not contain special safety mechanisms beyond their protocol specifications nor are they duplicated or spread over the PBRIDGE (in contrast to other I/Os, see I/O peripherals):

- FlexRay

- FEC
- M_CAN
- M_TTCAN

Software measures for the communication controllers (also called fault-tolerant communication layer) may contain E2E CRC data protection, sender identification, sequence numbering, and an acknowledgement mechanism.

PSI5 represents an exception to the replication of low bandwidth communications controllers. The module itself is replicated, but the two PSI5 modules share a single synchronization bit needed for staggered sync mode. This single bit is a single source of failures for both PSI5 modules if it triggers spurious synchronizations. The failure mode must be addressed by application level measures, which are in any case needed to cover the single transceiver used in such a configuration.

6.4.1.4.1 Disabling of communication controllers

In the event of a dangerous failure, the MCU offers the capability of disabling transmission of individual channels of communication controllers (such as CAN modules and FlexRay). This prevents the transmission of erroneous messages while preserving the capability of communicating over the diagnostic bus. Disabling outputs is controlled by resetting the SIUL2_MSCR*n*[SMC] for the pins that are associated with communication controllers where this feature is needed. See the "System Integration Unit Lite2 (SIUL2)" chapter for details on disabling communication signals.

The FCCU can drive a fault state on FI[0] via:

- Error pin reaction to a fault
- Software writing FCCU_CFG[FCCU_SET_CLEAR] = 01b

When the FCCU drives FI[0] to a fault state, the SIUL2 disables the output drivers of all pins whose SIUL2_MSCR*n*[SMC] = 0. This disables the possible transmission of erroneous messages. As long as the FCCU drives FI[0] to faulty state, the communication controllers' Tx pins are disabled and internal pull-up are enabled. As soon as FI[0] is driven to non-faulty state, the communication controllers' Tx pins are automatically enabled and state of internal pull-up/down restored.

The application should configure SIUL2_MSCR*n*[SMC] for pins that have active mapping of CAN modules (M_CAN, M_TTCAN) or FlexRay functionality and ensure that pin does not stay in undriven state.

Note

The FCCU uses an internal signal to disable the communication controller transmission, so transmission is disabled even when the FCCU cannot drive the FI[0] pin because the pin is configured as GPIO. If FI[0] is used as error input, externally pulling it down will disable the communication controller transmission if this event drives FCCU to a fault state.

Note

Disabling of communication controllers occurs on the Tx pins of the communication controller (for example, M_TTCAN, M_CAN, FlexRay) of the MCU even while the communication controller continues operating. Failed transmissions, when detected by the communication controller, may lead to additional error events.

6.4.1.5 Built-In Self-Tests (BIST)

The term BIST refers to the set of built-in hardware mechanisms that permits a device to test itself (typically at startup) in order to avoid the accumulation of latent faults, a requirement defined by the ISO 26262 standard.

Different types of BIST are implemented in the MPC5777M:

- LBIST for digital logic—managed by the Self-Test Control Unit.
- MBIST for memories—managed by the Self-Test Control Unit.
- MCU built-in mechanisms for testing analog peripherals—requires software intervention.

See the “Self-Test Control Unit (STCU2)” chapter for further details.

6.4.1.5.1 BIST during boot (off-line)

A device BIST is performed every time the device boots and is application transparent while the device is still under reset. BIST failure holds the device in reset, but the BIST continues running where possible and informs the system of the failure. Since LBIST and MBIST are destructive, a reset is performed for the tested module (memory or partition) before going into operation again. Application software starts when the BIST finishes without detecting a fault.

The boot time BIST comprises:

- Memory BIST for all RAMs and ROM
- Scan-based Logic BIST for digital logic, which is divided into multiple partitions that can be configured to be tested in parallel or sequentially to find the best time versus power consumption trade off.

6.4.1.5.2 Online Logical BIST (LBIST)

Though LBIST and MBIST are primarily intended to be run at startup under STCU2 control, the MCU allows software to trigger a memory self-test or LBIST of one partition during runtime. Since LBIST and MBIST are destructive, a reset is performed for the tested module (memory or partition) before going into operation again. This leads to a reset of the MCU or of those LBIST partitions. If a runtime LBIST requires a reset of the whole device, the LBIST result is accessible to software after the reset. If a reset occurs during LBIST, the LBIST is aborted. Output signals from each LBIST partition are set to a defined state when starting LBIST of that partition until the entire LBIST sequence completes. This has influence on the state of I/O signals. Table 6-3 and Table 6-4 below show the module groupings for each LBIST partition.

Note

When Scan mode is entered through LBIST, the flash memory must be in IDLE state, therefore Program or Erase operations must not be either active or suspended.

Table 6-3. LBIST partitions

LBIST partition	Instance name
LBIST 0 - HSM	HSM Instruction cache Data cache
LBIST 1 - IOP Core	IOP core (Core 2) Instruction cache STM_2 SWT_2
LBIST 2 - Peripheral Shell	SMPU_1 PBRIDGE_1 SWT_3 LINFLEX_[2,15] SENT_1 PSI5_1 IIC_1

Table continues on the next page...

Table 6-3. LBIST partitions (continued)

LBIST partition	Instance name
	DSPI_[2,3,5]
	CRC_1
	SARADC_[1:3,5:10]
	SDADC_[1,3,5,7,9]
	All CMUs
LBIST 3 - Computational Shell	SMPU_0
	PRAMC
	RAM
	PFLASHC
	Flash memory
	RAM overlay
LBIST 4 - Peripheral Shell Masters	eDMA
	DMAMUX
	DMA TCD RAM
	FEC
	RAM FEC
	FlexRay
	INTC
	FCCU
LBIST 5 - Peripherals	BAM ROM
	SARADC_[0,4,B]
	CRC_0
	IIC_0
	PSI5_0
	SENT_0
	CAN
	PBRIDGE_0
	MEMU
	WKPU
	NAR
	SPU
	PIT
	SDADC_[0,2,4,6,8]
	DSPI_[0,1,4,6,12]
	LINFlex_[0,1,14,16]
LBIST 6 - GTM	GTM
	GTM RAM
LBIST 7 - Safety Core	Master core (Core 0)

Table continues on the next page...

Table 6-3. LBIST partitions (continued)

LBIST partition	Instance name
	Instruction cache Data cache STM_0 SWT_0
LBIST 8 - Checker Core	Checker core (Core 0 - Checker) RCCU and delay logic
LBIST 9 - Computational Core	Computational Core (Core 1) Instruction cache Data cache STM_1 SWT_1

Table 6-4 shows the modules not covered by the LBIST partitions.

Table 6-4. Not in LBIST partitions

Functional group	Instance name
System	MC_CGM MC_ME MC_PCU MC_RGM SSCM SIUL2 IOMUX STCU2
Clocking	PLLs XOSC IRCOSC
Power	PMC ADC bandgap reference TSENS
Debug	JTAGC LFAST SIPI DCI JTAGM

6.4.1.6 Redundancy Control and Checker Units (RCCUs)

The RCCU compares a set of equivalent input signals provided by different sources and issues an alarm in case of a mismatch. For example, an RCCU compares the output of the checker core against the output of the main core and issues an alarm if they do not match. In case of a compare mismatch the fault stays stable until clearing the FCCU channel status.

6.4.1.7 FCCU and failure monitoring

The FCCU offers a hardware mechanism to aggregate error notifications and a configurable means to bring the device to a safe state. No CPU intervention is required for collection and control operation. Error indications are passed from the individual hardware components to the FCCU where the appropriate action is decided (according to the FCCU configuration).

6.4.1.7.1 External error indication

Failure of the MCU is signaled to one or two pins, FI[0] (ERROR0) and FI[1] (ERROR1). FI[0] can also serve as an error input mechanism (see "Fault Collection and Control Unit (FCCU)" chapter, and "FCCU configuration" section of the "Device configuration" chapter, for details on the fault output signals). Both pins can be multiplexed with other functions with different defaults.

The error indication on pins FI[0] and FI[1] is controlled by SIUL2 and FCCU. The mapping of the FCCU output signals on the relevant pins and the associated electrical characteristics are enabled by the SIUL2_MSCR[*n*] register (see table "SIUL2_MSCR0 - SIUL2_MSCR511 field description" in the "System Integration Unit Lite2 (SIUL2)" chapter for signal mapping details). Once the FCCU is mapped onto the pin, the logic level is driven by the FCCU.

FI[0] is controlled by the FCCU at startup by default, but the state of FI[1] is like other I/O signals (see table "Pin/ball startup and reset states" in chapter "Signal Description" for default states of FI[0] and FI[1]).

Since it is possible to reset FCCU and SIUL2 independently, the behavior of FI[0] and FI[1] is the result of the combined state of SIUL2_MSCR[*n*] and the FCCU. (see section "Reset interface" in chapter "Fault Collection and Control Unit (FCCU)" for FCCU reset details, and chapter "Reset and Boot" for the state of the FCCU relative to the system during different reset phases).

FCCU_STAT[ESTAT] flags whether the FCCU is in an error state. This flag can be set by software to a '1' (fault) or '0' (operational) when the FCCU is in operational state. FCCU_STAT[PSTAT] mirrors the physical state of the *Fn* external pin's value, though this may differ from the logical state if a toggling protocol is used.

6.4.1.7.2 Failure handling

The FCCU is an autonomous module that manages the different reactions configured for each failure source. Overall failure reaction time includes error detection, processing, and indication. In this interval, the MPC5777M may provide wrong results to the system.

Failure sources include:

- All failure indication signals from modules within the MCU
- Control logic and signals monitored by the FCCU itself (see chapter "Fault Collection and Control Unit (FCCU)").
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Note that software can also directly influence the state of the F*n* pins.
- External failure input.

Available failure reactions are:

- Assertion of an interrupt (maskable or non-maskable)
- Resetting the MCU
- Changing the state of the failure indication pins, ERROR*n*
- Disabling the transmission capabilities of communication controllers (FlexRay, CAN, FEC) (Note: only possible in conjunction with changing the state of the failure indication pins)
- No reaction

Software can read the failure source that caused a fault, either before or after a functional reset (the condition indicators are not volatile). Software can also reset the failure, but the external failure indication remains for a configurable minimum time. If necessary, software can also reset the MCU

6.4.1.8 Operational interference protection

Being a multi-master system, MPC5777M provides safety mechanisms to prevent non-safety masters from interfering with the operation of the Safety Core, as well as mechanisms to handle the concurrent execution of software with different (lower) ASIL. Interference freedom mechanisms are arranged in a hierarchical memory protection schema including:

- MPUs
- PBRIDGEs
- Register protection.

There are two Memory Protection Unit levels included in the MPC5777M. The Core Memory Protection Unit (CMPU) is a mechanism included in each core (except HSM core) to protect address ranges against access by software developed according to lower ASIL. The CMPU is typically used by the operating system to ensure inter-task interference protection.

The second MPU level is provided by the System MPUs (SMPU) located in each XBAR. The MPUs prevent access of different bus masters to address ranges and are typically used by the safety application to prevent NoSaMos (such as the DMA or the Performance Core) to access the application's safety-relevant resources.

Furthermore, the PBRIDGE can restrict read and write access to individual I/O modules based on the origin of the access and its state (user mode/supervisor mode).

Finally, the register protection included allows individual registers to be locked against any manipulation without unlocking.

These safety mechanisms are further described in the "System Memory Protection Unit (SMPU)" chapter.

6.4.1.9 FCCU supervision (FOSU)

As the FCCU is a central component in reacting to errors, it is itself supervised even though an error in it can only cause a latent failure. This supervision is provided by the FCCU Output Supervision Unit (FOSU). The FOSU receives failure indications at the same time as the FCCU. Unless the respective failure is switched off, the FOSU observes the outputs of the FCCU (IRQ, RESET, F*n*). If the FCCU does not react within a predefined interval as specified in the "FCCU configuration" section of the "Device configuration" chapter, the FOSU assumes the FCCU has failed and causes a destructive reset.

The FOSU does not require any configuration by software.

6.4.2 Dual-core lockstep

The MPC5777M duplicates its safety-relevant processing elements and compares their operation in Lockstep mode (LSM). This Safety core consists of two cores, Checker Core_0 and Main Core_0, and as far as software is concerned they behave as one core. Main Core_0 is the main execution core of the pair, where Checker Core_0 follows the execution of the Main Core in lockstep.

The replicated processing elements contain:

- Core
- Cache control
- Local memory control
- Core MPU
- Core System Bus Interface, including e2eECC logic

Together each set of replicated elements forms a channel (for example, the Main channel and the Checker channel).

Elements that are not replicated are:

- Performance or I/O cores
- Cache and local memory arrays
- INTC
- Interconnect (XBAR)
- RAM controller
- Flash memory controller
- eDMA controller

Note

This is not a complete list of all non-replicated elements. The elements listed are sometimes replicated in similar lockstep architectures.

Equivalent operation of replicated resources is supervised by comparators on all functional signals leaving the channels for the rest of the MCU. Any operational deviations between the supervised signals causes the FCCU to be notified of the discrepancy.

The Checker Core does not have a direct connection to the XBAR. All of the outputs of Checker Core_0 that target the XBAR (as well as any other non-duplicated resource, like local memories) end in an RCCU for verification, and all the inputs to Checker Core_0 from the XBAR are split off from the Main Core_0 XBAR inputs.

An abstract view of the implementation including delays and RCCUs is shown in Figure 6-2, but does not show local memory.

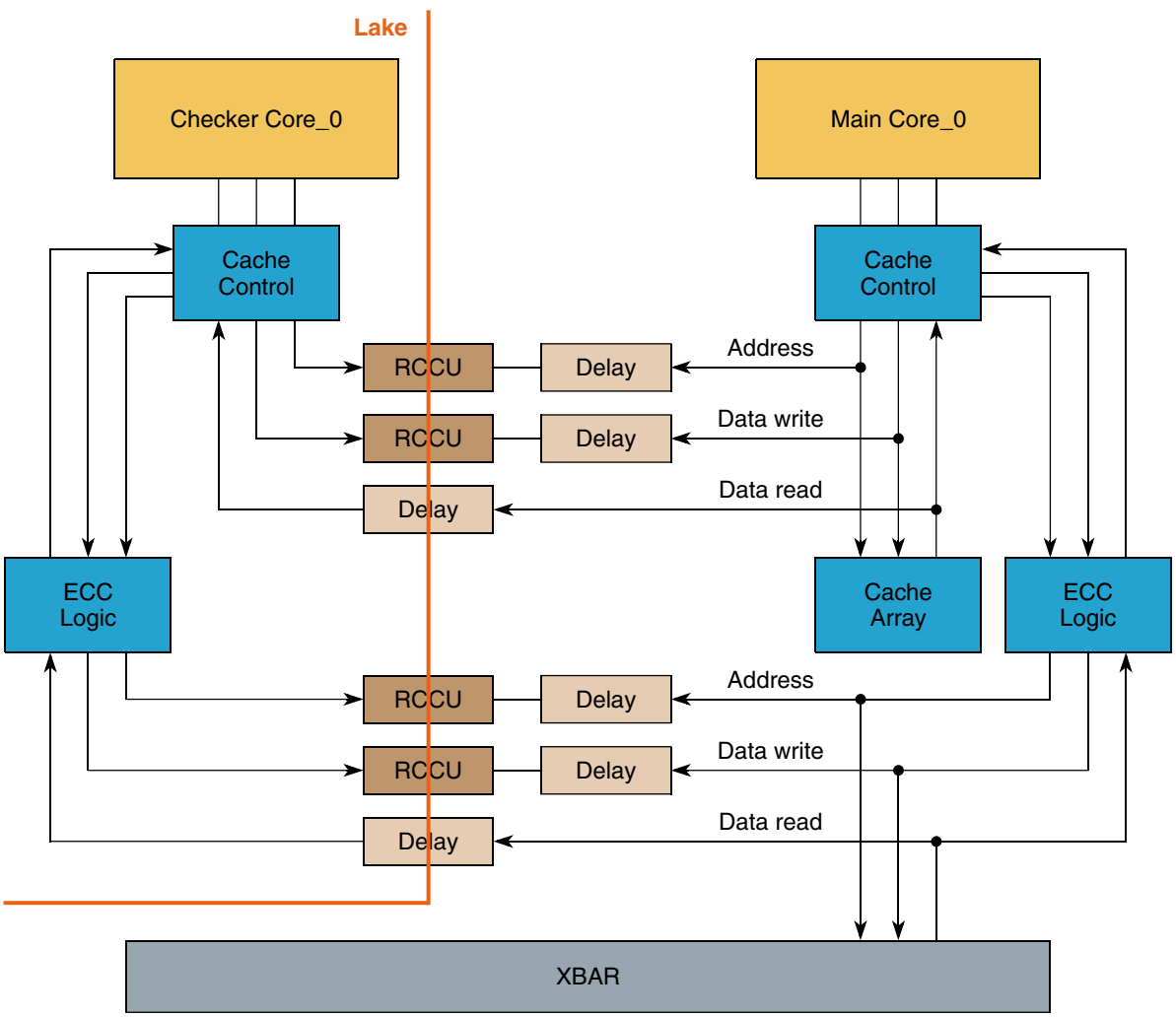


Figure 6-2. Dual core concept

A delay of two clock cycles exists between the Main and Checker channel to reduce their susceptibility to CCFs caused by power and clock disturbances. Outputs of the Main Core (such as XBAR addresses) are registered and delayed before being forwarded to the RCCUs, while inputs signals (such as XBAR read data) are registered and delayed before being fed to the Checker channel.

LSM can be disabled during boot by reprogramming the Flash memory (UTEST Miscellaneous DCF Client) or on-line by disabling the checker but not the main Core_0[1]. LSM is also automatically disabled when Core_0 enters debug mode and automatically reenabled when it leaves it. If LSM is disabled, the RCCUs are disabled and a signal is sent to the FCCU. If SW enables the checker core by writing in MC_ME - MC_ME_CCTL register while LSM is disabled (not recommended), this does not enable LSM and the RCCUs do not generate any errors in case of mismatch. Thus no user-initiated dynamic enabling of LSM is possible. Furthermore, it is not a valid operation to stop main core_0 only or to reset one of the cores only. If the cores are out of lock, a destructive or functional reset is needed to reestablish LSM.When safety core enters debug mode, the signaling of potential mismatches by RCCUs (FCCU channels #10 and #11) is disabled. RCCU signaling is automatically re-enabled when safety core leaves debug mode. If LSM or the RCCU signaling is disabled a signal is sent to the FCCU (FCCU channels #51). A reset is needed to reestablish LSM. Also, in contrast to the MPC564xL , no Decoupled Parallel Mode (DPM) is available (Main Core_0 and Checker Core_0 cannot operate as two independent cores running different software). Main Core_0 is able to operate as an independent core if Checker Core_0 is disabled. The status of LSM for the Safety core can be verified by checking the appropriate FCCU register.

6.4.3 Common Cause Failure measures

Various measures are included to prevent CCFs from endangering the effectiveness of the replication of the Safety Core or of the peripherals. These measures include physical separation of the components on the die, routing restrictions and supervision of clock, power, temperature, test and debug signals. In general these measures are independent from the software.

There are also configuration registers that can affect the execution of the MCU's safety function when changed and simultaneously disable the respective safety mechanism. These are either protected against bit flips or those flips are detected by independent measures. These same registers are also protected against accidental software writes by employing the register protection described above.

1. Disabling both cores in a single mode transition will not disable LSM.

6.4.4 ECC

Error correcting codes are used for end-to-end protection from cores to system storage as well as for individual protection of peripheral RAMs.

6.4.4.1 ECC for storage

All storage (system RAM, local memory, Flash memory, peripheral RAM) used in normal operation is protected by ECC with SECDED (Single Error Correct and Double Error Detect) with the following exceptions:

- The caches in the cores are protected by EDC (Error Detection Code), which only detects errors.
- FlexRay's schedule information RAM is protected by EDC only.

A list showing the implementation of RAMs with ECC (including address protection) and for the MPC5777M is shown in Table 6-5.

Table 6-5. ECC RAM implementations

Location	Memory	Memory classification	ECC	Address in ECC	Muxing factor	Reacts to uncorrectable error
Core_0 (Safety Core) (Master and Checker cores)	I-Mem	System RAM	SEC/DED	Y	8	Y
	D-Mem	System RAM	SEC/DED	Y	8	Y
	I-Cache	System RAM	EDC	Y	4	Y
	D-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
	DTAG	System RAM	SEC/DED	Y	4	Y
Core_1	I-Mem	System RAM	SEC/DED	Y	8	Y
	D-Mem	System RAM	SEC/DED	Y	8	Y
	I-Cache	System RAM	EDC	Y	4	Y
	D-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
	DTAG	System RAM	SEC/DED	Y	4	Y
Core_2	I-Mem	System RAM	SEC/DED	Y	8	Y
	D-Mem	System RAM	SEC/DED	Y	8	Y
	I-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
HSM	PRAM	Other	SEC/DED	Y	8	N

Table continues on the next page...

Table 6-5. ECC RAM implementations (continued)

Location	Memory	Memory classification	ECC	Address in ECC	Muxing factor	Reacts to uncorrectable error
	I-Cache	Other	EDC	Y	4	N
	ITAG	Other	EDC	Y	4	N
	C3	Other	No	N	4	N
FlexRay	DRAM	Peripheral	SEC/DED	N	4	Y
	LRAM	Peripheral	EDC	N	4	N
DPLL-2	Peripheral	SEC/DED	Y	8	N	
GTM	FIFO0 and FIFO1	Peripheral	SEC/DED	Y	8	N
	(MCSi)-RAM0, i=0~5	Peripheral	SEC/DED	Y	8	N
	(MCSi)-RAM1, i=0~5	Peripheral	SEC/DED	Y	4	N
	DPLL-1A	Peripheral	SEC/DED	Y	4	N
	DPLL-1B	Peripheral	SEC/DED	Y	4	N
	DPLL-2	Peripheral	SEC/DED	Y	16	N
TTCAN	TTCAN	Peripheral	SEC/DED	N	16	Y
NAR	NAR	Other	No	—	4	N
Platform/DMA	DMA	Peripheral	SEC/DED	Y	4	Y
Platform	Overlay	System RAM	SEC/DED	Y	4	Y
Platform	SRAM	System RAM	E2E-ECC	Y	16	Y
Platform	FEC/FIFO	Peripheral	SEC/DED	Y	4	N
Platform	FEC/MIB	Peripheral	SEC/DED	Y	4	N

Table 6-5 shows that some memories, particularly system storage, use an ECC computed over data and address to detect data and addressing faults (no/wrong/multiple selection). In addition, these same memories include dedicated measures against addressing and control faults (such as address/control feedback).This is different for storage related to PeMos. PeMos generally use an ECC without address error protections and do not include additional measures to detect them. All ECC calculations for RAMs in PeMos are accomplished by logic outside of the RAM, not in the RAM block.

An exception to this is the GTM's RAMs. They include addressing failure detection measures in the ECC. In addition, the individual RAM ports of the GTM do not share common parts and connect to individual (non-interfering) RAM arrays.

6.4.4.2 All-x words and ECC

There is a special case for legal ECC values in the MPC5777M. Memory entries that are all zeros (All-0) or all ones (All-1), including the ECC parity bits, are not legal for memory that is checked by ECC. The flash memory allows All-1, corresponding to the status of an erase block, as a valid codeword.

Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses, All-0 or All-1 may be legal.

All-0 and All-1 memory content is indicated in different ways. For memories that do not include address into the ECC calculation, All-0 and All-1 will be uncorrectable errors.For all memories that include address into the ECC code-bit calculation, since the ECC checkbits depend on the address, it is not possible to generate an uncorrectable error indication for all the possible addresses. Therefore, an All-x content may result in a correctable error.

Note

For flash memory, additional dedicated safety mechanisms exist to detect failures that have the potential of leading to an All-1 word (see "Embedded Flash Memory" chapter, for more details on flash memory safety mechanisms).

6.4.4.3 ECC failure handling

Single-bit and double-bit errors (correctable and uncorrectable errors) are signaled to the FCCU unless filtered by the MEMU . The MEMU (see Memory Error Management Unit (MEMU)) may filter ECC error notification for known ECC error addresses (known permanent correctable errors). Actual implementation will signal errors, not to the FCCU, but to the MEMU, which filters and then forwards unfiltered notifications in an aggregated manner to the FCCU.

6.4.4.4 Memory Error Management Unit (MEMU)

The MEMU is responsible for the collection and reporting of error events associated with ECC logic used on SRAM (system RAMs in this context are those RAMs with "System RAM" memory classification as in Table 6-5), peripheral RAM, and flash memory. When ECC error events occur, the MEMU receives an error signal that causes an event to be recorded, and possibly sets corresponding error flags that are reported to the FCCU .

The MEMU stores the addresses of ECC errors that have occurred in a table as follows:

- Uniqueness of the addresses in the table is ensured. For example, if an address is already stored in the MEMU's table, correctable errors at that address will no longer be reported to the FCCU.
- Software can read and modify the MEMU table and remove individual entries (by marking them as invalid). For example, this allows software to invalidate a new entry in the MEMU table and wait for a repeat occurrence of the ECC error indicating a permanent error in memory.
- If the MEMU table overflows, an additional signal (instead of the ECC error signal) is sent to the FCCU.

As the MEMU aggregates address information from several sources, it might not be able to process all simultaneously arriving reports. This is called a simultaneous overflow and is signaled to the FCCU as a buffer overflow (see Memory Error Management Unit (MEMU) chapter for more information).

6.4.4.4.1 Interface to ECC units

The MEMU receives data according to the following interface per ECC unit connected:

- Whether the error is a Single-Bit-Error (SBE) or Uncorrectable Error (UCE) type
- Address of the memory where the error occurred
- the (optional) ECC-syndrome or MBIST bad-bit information
- A configuration specifying whether the ECC unit is attached to a vital memory (SRAM, local memory, and system RAM of Safety Core), to flash memory, or to a peripheral RAM

Not all ECC units provide a syndrome. If an ECC unit does not provide the syndrome, FFh is stored in the MEMU error table instead of the syndrome. If an error is signaled, it is compared against all errors known for that storage:

Note

If a previously known error has been marked invalid by software, no comparison against the address will occur.

- If no entry of that address is already in the buffer, it is to be added into the appropriate table depending on the SBE versus UCE indicator
- If there is no free entry left in that buffer, the overflow flag is set.

If a valid entry of that address is in the buffer and If the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer. In case the same error is reported by ECC which reports the syndrome and additionally by ECC which does not, the reporting order becomes important.

- If the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer.If ECC which provides the syndrome reports first, the second reporting without syndrome is discarded.
- If ECC which does not provide the syndrome reports first, a second entry with the syndrome is stored in the table.

Chapter 7
Device Configuration

7.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- Module block diagrams showing immediate connections within the device
- Specific module-to-module interactions not necessarily discussed in the individual Module chapters
- Links for more information

7.2 Core modules

The MPC5777M has cores in three distinct modules:

- HSM (described in the HSM chapter of the *Security Reference Manual*)
- GTM (described in the *GTM Reference Manual*)
- Main platform has four separate cores that perform various computational and control functions as described below.

Table 7-1. MPC5777M core modules

Instance	Description
Main Core_0	The main computational shell consists of Main Core_0 and Main Core_1, both of which use an e200z710n3 core. These two cores are used for general computational functions.
Main Core_1	
Checker Core_0s	A third core, referred to as Checker Core_0s, uses an e200z709 core, which is a true subset of the e200z710n3 core. When enabled, Checker Core_0s operates in lock step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to insure that Main Core_0 is executing correctly. There is no lock step function associated with Main Core_1.
Peripheral Core_2	The fourth processor, Peripheral Core_2, employs an e200z425Bn3 core and is generally used to control various I/O modules.

For more information, please see Core Complex Overview.

7.2.1 Core Reset Settings

Table 7-2 shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 7-2. Reset settings for e200z resources

Resource	System reset setting
Program Counter	p_rstbase[0:29] || 2'b00[1]
GPRs	Unaffected[2]
CR	Unaffected[2]
BUCSR	0x0000_0000
CSRR0	Unaffected[2]
CSRR1	Unaffected[2]
CTR	Unaffected[2]
DAC1	0x0000_0000[3]
DAC2	0x0000_0000[3]
DAC3	0x0000_0000[3]
DAC4	0x0000_0000[3]
DBCR0	0x0000_0000[3]
DBCR1	0x0000_0000[3]
DBCR2	0x0000_0000[3]
DBCR4	0x0000_0000[3]
DBCR5	0x0000_0000[3]
DBCR6	0x0000_0000[3]
DBCR7	0x0000_0000[3]
DBCR8	0x0000_0000[3]
DBSR	0x1000_0000[3]
DDAM	0x0000_0000[3]
DDEAR	Unaffected[2]
DEAR	Unaffected[2]
DEVENT	0x0000_0000[3]
DSRR0	Unaffected[2]
DSRR1	Unaffected[2]
DVC1	Unaffected[2]
DVC2	Unaffected[2]
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000[3]

Table continues on the next page...

Table 7-2. Reset settings for e200z resources (continued)

Resource	System reset setting
IAC2	0x0000_0000[3]
IAC3	0x0000_0000[3]
IAC4	0x0000_0000[3]
IAC5	0x0000_0000[3]
IAC6	0x0000_0000[3]
IAC7	0x0000_0000[3]
IAC8	0x0000_0000[3]
IVPR	Unaffected[2]
LR	Unaffected[2]
L1CFG0, L1CFG1[4]	—
L1CSR0, 1	0x0000_0000
L1FINV0, 1	0x0000_0000
MAS1	Unaffected[2]
MAS2	Unaffected[2]
MAS3	Unaffected[2]
MAS4	Unaffected[2]
MCAR	Unaffected[2]
MCSR	0x0000_0000
MCSRR0	Unaffected[2]
MCSRR1	Unaffected[2]
MMUCFG4	—
MPU0CSR0	0x0000_0000
MPU0CFG[4]	—
MSR	0x0000_0000
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_00 || p_cpuid[0:7][1]
PVR[4]	—
SPEFSCR	0x0000_0000
SPRG0	Unaffected[2]
SPRG1	Unaffected[2]
SPRG2	Unaffected[2]
SPRG3	Unaffected[2]
SRR0	Unaffected[2]
SRR1	Unaffected[2]
SVR[4]	—
XER	0x0000_0000

1. The levels that determine these bits are set by the factory and may change between revisions of the device.

   Main_Core_0: p_cpuid[0:7] = 0x00

Main_core_1: p_cpuid[0:7] = 0x01

Peripheral_Core_2: p_cpuid[0:7] = 0x02

2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. Reset by processor reset **external PORESET** if DBCR0[EDM]=0, as well as unconditionally by **POR**.
4. Read-only registers.

7.2.2 Special Purpose Register Summary

PowerISA 2.06 and implementation-specific SPRs for the e200z710n3, e200z709, and e200z425Bn3 cores are listed in the following table. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR register may be read or written with the **mfspr** and **mtspr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in the following table.

Table 7-3. Special Purpose Registers

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/ Clear[1]	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No

Table continues on the next page...

Table 7-3. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1[2]	318	R/W	Yes	No
DVC2	Data Value Compare 2[2]	319	R/W	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCNT	Debug Counter register	562	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/ Clear[3]	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U[4]	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U[4]	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes
EDBRAC0	External debug resource allocation control register 0	638	Read only	Yes	Yes

Table continues on the next page...

Table 7-3. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MPU0CSR0	MPU0 configuration register	1014	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using *mfspr RT,DBSR*. The Debug Status Register cannot be directly written to. Instead, bits in the Debug Status Register corresponding to '1' bits in GPR(RS) can be cleared using *mtspr DBSR,RS*.
2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. The Machine Check Syndrome Register can be read using *mfspr RT,MCSR*. The Machine Check Syndrome Register cannot be directly written to. Instead, bits in the Machine Check Syndrome Register corresponding to '1' bits in GPR(RS) can be cleared using *mtspr MCSR,RS*.
4. The 64-bit DVC registers are accessed by using the DVC1,2 spr #s for the lower 32 bits, and the DVC1,2U spr #s for the upper 32 bits.

7.3 System modules

7.3.1 SIUL2 configuration

The System Integration Unit Lite 2 (SIUL2) controls:

- MCU pad configuration
- Ports
- General-purpose input and output (GPIO) signals
- External interrupts with trigger event configuration

For details please refer System Integration Unit Lite2.

7.3.2 Semaphores2 (SEMA42)

The peripheral platform shell includes a memory-mapped module (SEMA42) that provides

- Robust hardware support needed in multi-core systems for implementing semaphores
- Asimple mechanism to achieve "lock/unlock" operations via a single write access

The hardware semaphore module is an architecture-neutral solution that provides sixteen hardware-enforced gates for MPC5777M as well as other useful system functions related to the gating mechanisms. The capabilities provided by the hardware semaphore module are currently utilized in FSL's dual-core AUTOSAR® implementations and its use is expected to continue in future multi-core ports. The result is a second-generation module implementation known as Semaphores2 or SEMA42.

For details refer to Semaphores2 (SEMA42).

7.3.3 Crossbar switch configuration

This device contains two crossbar switch modules. Figure 7-1 illustrates that they are connected via a Cross-lake, which is composed of intelligent bridging bus gaskets. These bus gaskets handle the traffic that crosses between the shells and do not require any interaction with the user. This dual-crossbar architecture allows all masters to access all slaves across the computational shell and peripheral shell.

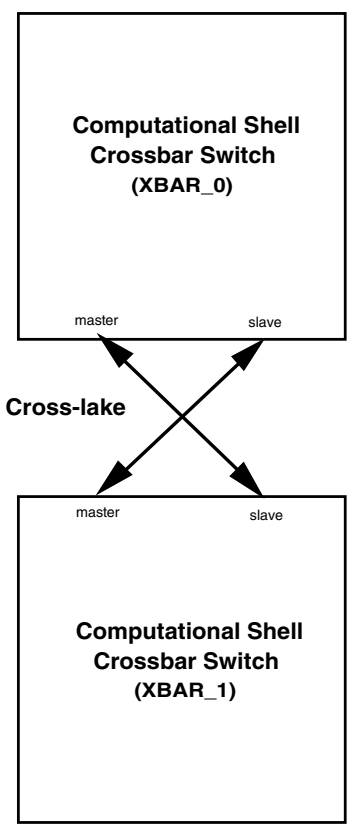


Figure 7-1. Crossbar switches diagram

The following sections explain how each crossbar switch is configured on this device.

7.3.3.1 Computational shell crossbar switch (XBAR_0) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

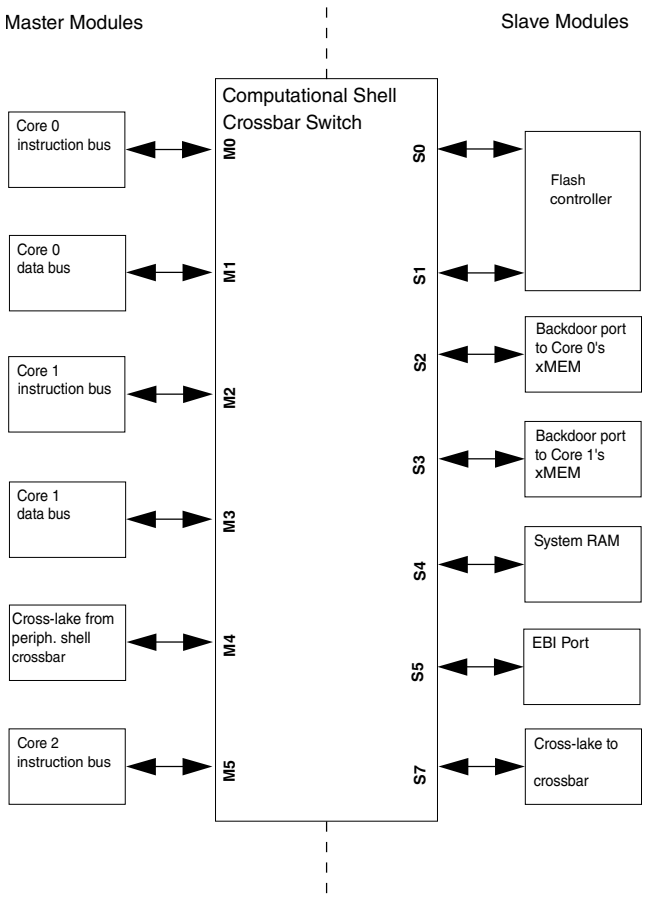


Figure 7-2. Computational shell's crossbar switch integration

Table 7-4. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	XBAR chapter
System memory map	—	Memory Map
Clocking	—	Clocking
Crossbar switch master	Cores	Core modules
Crossbar switch slave	Flash memory controller	Flash memory controller
Crossbar switch slave	Processor core local memory	Processor core local SRAM
Crossbar switch slave	System RAM	System SRAM

7.3.3.1.1 Computational shell crossbar switch master assignments

This device contains six masters connected to the computational shell's crossbar switch.

The master assignment is shown in Table 7-5.

Table 7-5. Computational shell crossbar switch master assignments_a

Master module	Physical master port number
Core 0 instruction bus	0
Core 0 data bus	1
Core 1 instruction bus	2
Core 1 data bus	3
Peripheral shell crossbar switch	4
Core 2 instruction bus	5

7.3.3.1.2 Computational shell crossbar switch slave assignments

This device contains seven slaves connected to the computational shell's crossbar switch.

The slave assignment is shown in Table 7-6.

Table 7-6. Computational shell crossbar switch slave assignments

Slave module	Physical slave port number
Flash memory controller port 0	0
Flash memory controller port 1	1
Backdoor port to Core 0's xMEM	2
Backdoor port to Core 1's xMEM	3
System RAM	4
EBI port	5
Peripheral shell crossbar switch	7

Access to the flash memory via slave ports 0 and 1 is routed based on the master. The master-to-flash memory port routing has been assigned in a way to balance data traffic, with attention paid to facilitate core instruction fetches. The following table shows the flash memory port assignments.

Table 7-7. Flash memory port assignments

Flash memory slave port 0	Flash memory slave port 1
Core 0 instruction bus (M0)	Core 0 data bus (M1)
Core 1 data bus (M3)	Core 1 instruction bus (M2)

Table continues on the next page...

Table 7-7. Flash memory port assignments (continued)

Flash memory slave port 0	Flash memory slave port 1
Peripheral shell crossbar switch (M4)	Core 2 instruction bus (M5)

7.3.3.1.3 PRS register reset values

The computational shell's XBAR_0 PRSn registers reset to 0x0030_1425. Note that these registers are read-only.

7.3.3.1.4 CRS register reset values

The computational shell's XBAR_0 CRSn registers reset to 0x003F_0100.

7.3.3.2 Peripheral shell crossbar switch (XBAR_1) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

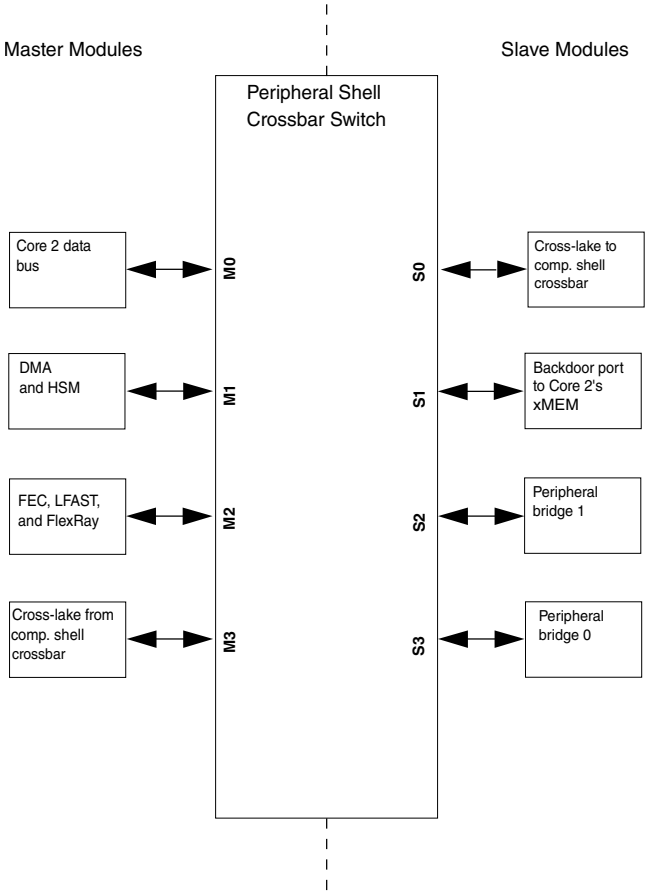


Figure 7-3. Crossbar switch integration

Table 7-8. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	XBAR chapter
System memory map	—	Memory Map
Clocking	—	Clocking
Crossbar switch master	Cores	Core modules
Crossbar switch master	DMA and HSM	DMA Controller configuration
Crossbar switch master	FEC, LFAST, and FlexRay	FEC interface selection, LFAST and SIPI, FlexRay Configuration
Crossbar switch slave	Processor core local memory	Processor core local SRAM
Crossbar switch slave	Peripheral bridge	Peripheral Bridge

7.3.3.2.1 Peripheral shell crossbar switch master assignments

This device contains four masters connected to the peripheral shell's crossbar switch.

The master assignment is shown in Table 7-9.

Table 7-9. Peripheral shell crossbar switch master assignments_a

Master module	Physical master port number
Core 2 data bus	0
DMA and Hardware Security Module (HSM)	1
FEC, LFAST, and FlexRay	2
Cross-lake master	3

7.3.3.2.2 Peripheral shell crossbar switch slave assignments

This device contains four slaves connected to the peripheral shell's crossbar switch.

The slave assignment is shown in Table 7-10.

Table 7-10. Peripheral shell crossbar switch slave assignments_a

Slave module	Physical slave port number
Computational shell crossbar switch	0
Backdoor port to Core 2's xMEM	1
Peripheral bridge 1	2
Peripheral bridge 0	3

7.3.3.2.3 PRS register reset values_a

The peripheral shell's XBAR_1 PRS*n* registers reset to 0x0000_3102.

7.3.3.2.4 CRS register reset values_a

The peripheral shell's XBAR_1 CRS*n* registers reset to 0x000F_0100.

7.3.4 System Memory Protection Unit (SMPU) configuration

This chip contains three instances of the System Memory Protection Unit (SMPU):

- SMPU_HSM for the Hardware Security Module
- SMPU_0 for the Computational Shell
- SMPU_1 for the Peripheral Shell

The following sections explain how each SMPU is configured on this chip. See the *Microcontroller Security Reference Manual* for configuration details of SMPU_HSM.

7.3.4.1 SMPU0 configuration

The following sections explain how SMPU_0 and SMPU_1 are configured on this chip. For a comprehensive description of the module itself, see the chapter for that module.

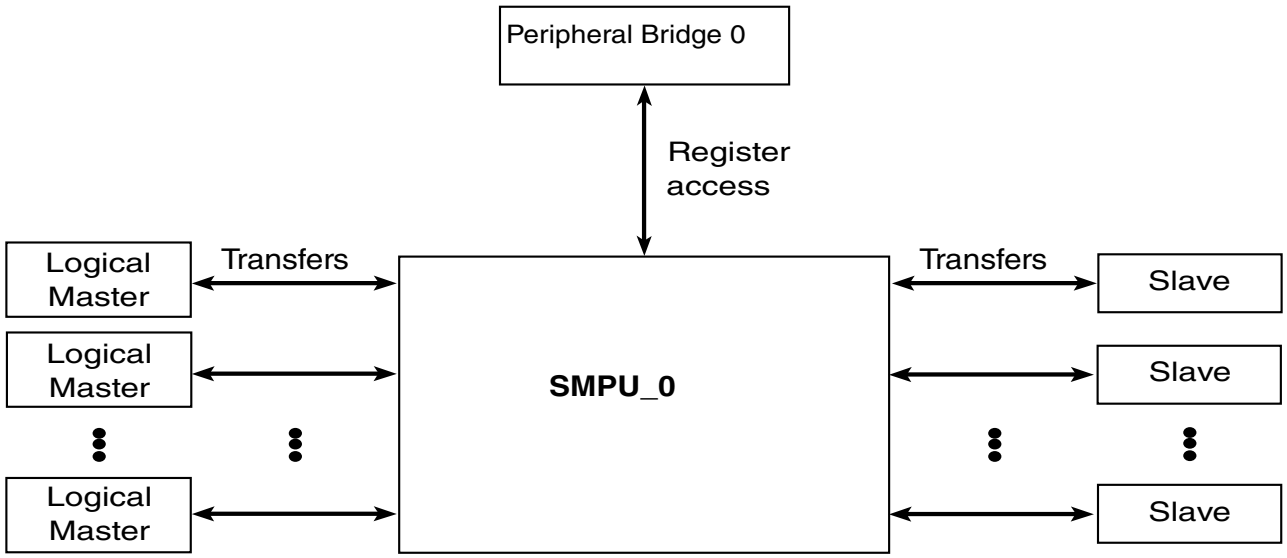


Figure 7-4. SMPU0 configuration

Table 7-11. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	System Memory Protection Unit (SMPU)
System memory map	—	Memory Map
Clocking	—	Clocking
Power management	—	Power management
Logical masters	—	SMPU0 logical bus master assignments
Region descriptors	—	Number of region descriptors supported by SMPU0
RGD_WORD2 formats	—	RGD_WORD2 formats supported by SMPU0
Master Access Permissions	—	Master Access permissions for SMPU0

7.3.4.1.1 SMPU0 logical bus master assignments

The logical bus master assignments for the SMPU0 are:

Table 7-12. SMPU0 logical bus master assignments_a

SMPU logical bus master number	Bus master
0	Core_0
1	Core_1
2	Core_2
3	DMA_0
4	Ethernet
5	FlexRay_0
6	SIPI_1
7	SIPI_0/LFAST
8	Core_0 Debug
9	Core_1 Debug
10	Core_2 Debug
11	DMA_1
12	Reserved
13	FlexRay_1
14	HSM
15	NAR

7.3.4.1.2 Number of region descriptors supported by SMPU0

This instance of the SMPU module supports a maximum of twelve region descriptors.

7.3.4.1.3 RGD_WORD2 formats supported by SMPU0

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.4.1.4 Master Access permissions for SMPU0

The only masters allowed to access the SMPU0 registers are HSM, Core 2, Core 1, and Core 0. It is also required that the core be in supervisor mode.

7.3.4.2 SMPU1 configuration

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

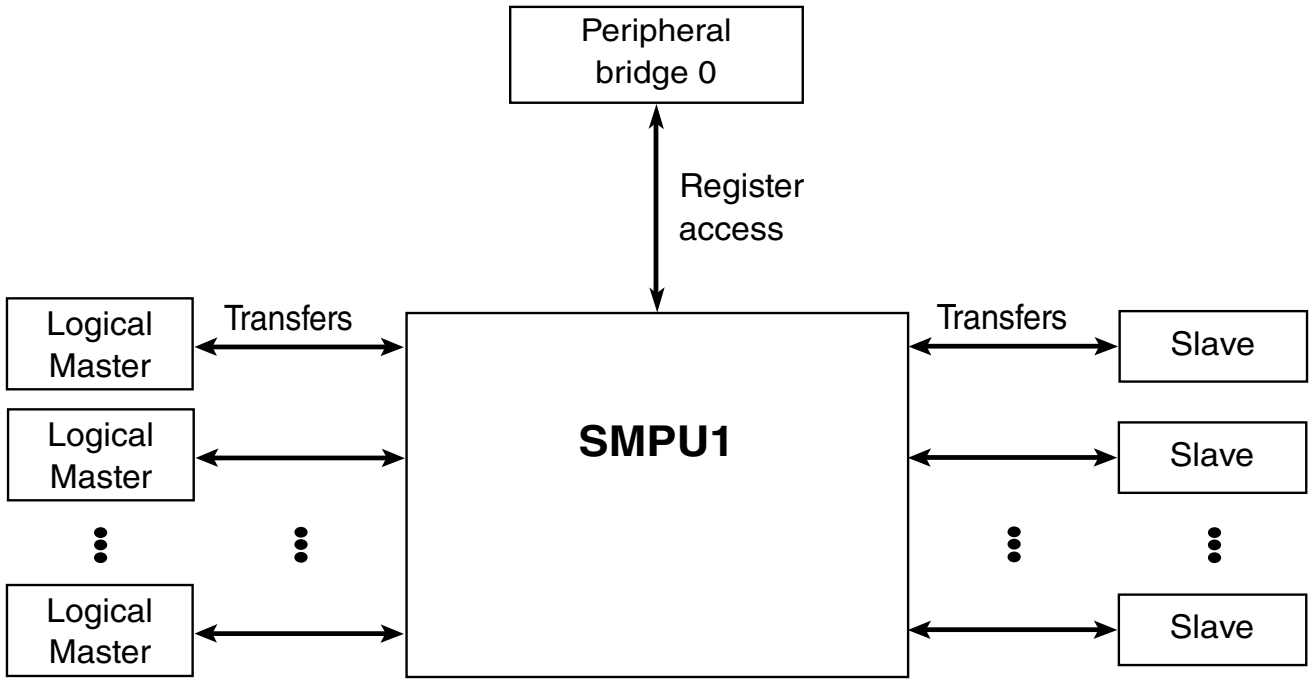


Figure 7-5. SMPU1 configuration

Table 7-13. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	System Memory Protection Unit (SMPU)
System memory map	—	Memory Map
Clocking	—	Clocking
Power management	—	Power management
Logical masters	—	SMPU1 logical bus master assignments
Region descriptors	—	Number of region descriptors supported by SMPU1
RGD_WORD2 formats	—	RGD_WORD2 formats supported by SMPU1
Master Access Permissions	—	Master Access permissions for SMPU1

7.3.4.2.1 SMPU1 logical bus master assignments

The logical bus master assignments for the SMPU1are:

Table 7-14. SMPU1 logical bus master assignments_a

SMPU logical bus master number	Bus master
0	Core_0
1	Core_1
2	Core_2

Table continues on the next page...

Table 7-14. SMPU1 logical bus master assignments_a (continued)

SMPU logical bus master number	Bus master
3	DMA_0
4	Ethernet
5	FlexRay_0
6	SIPI_1
7	SIPI_1/LFAST
8	Core_0 Debug
9	Core_1 Debug
10	Core_2 Debug
11	DMA_1
12	Reserved
13	FlexRay_1
14	HSM
15	NAR

7.3.4.2.2 Number of region descriptors supported by SMPU1

This instance of the SMPU module supports a maximum of eight region descriptors.

7.3.4.2.3 RGD_WORD2 formats supported by SMPU1

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.4.2.4 Master Access permissions for SMPU1

The only masters allowed to access the SMPU1 registers are HSM, Core 2, Core 1, and Core 0. It is also required that the core be in supebe in supervisor mode.

7.3.5 Peripheral bridge configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

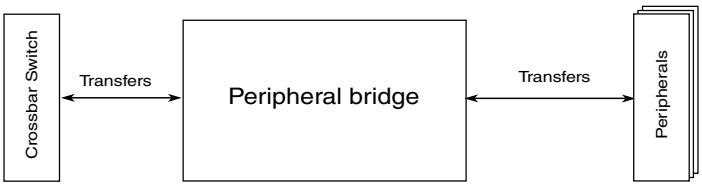


Figure 7-6. Peripheral bridge configuration

Table 7-15. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	Peripheral Bridge
System memory map	—	Memory Map
Clocking	—	Clocking
Crossbar switch	Crossbar switch	Crossbar switch configuration

7.3.5.1 Number of peripheral bridges

This device contains two identical peripheral bridges.

7.3.5.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See Memory Map" for the memory slot assignment for each module.

7.3.5.3 MPR registers

Each of the two peripheral bridges supports up to 16 crossbar switch masters, each assigned to a MPROT*n* field in the MPRA-B registers. However, fewer are supported on this device. See Crossbar switch configuration for details of the master port assignments for this device.

7.3.5.4 PACR/OPACR registers

The peripherals attached to the peripheral bridges each are assigned to a PACR*n* or OPACR*n* field within the PACRA-H/OPACRA-AF registers. See Memory Map chapter for details of the peripheral slot assignments for this device. Unused PACR*n* fields are reserved.

7.3.6 Platform Configuration Module (PCM)

The Platform Configuration Module contains three miscellaneous configuration registers for the device. Currently, the configuration registers are related to the operation of the FEC and intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000.

7.3.6.1 FEC Burst Optimization Master Control Register (FBOMCR)

Offset: 00h Access: Supervisor read/write

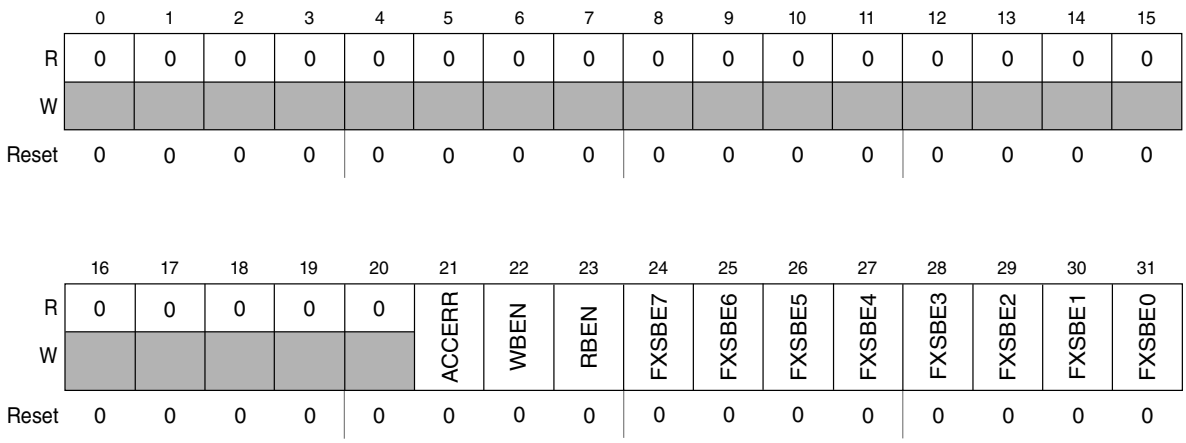


Figure 7-7. FEC Burst Optimization Master Control Register (FBOMCR)

Table 7-16. FBOMCR field description

Field	Description
0–20	Reserved
21 ACCERR	Accumulate error This bit determines whether an error response for the first half of the write burst is accumulated to the second half of the write burst or discarded. In order to complete the burst, the FEC interface to the system bus responds by indicating that the first half of the burst completed without error before it actually writes the data so that it can fetch the second half of the write data from the FIFO. When actually written onto the system bus, the first half of the write burst can have an error. Because this half initially responded without an error to the FIFO, the error is discarded or accumulated with the error response for the second half of the burst. 0 Any error to the first half of the write burst is discarded. 1 Any actual error response to the first half of the write burst is accumulated in the second half's response. In other words, an error response to the first half will be seen in the response to the second half, even if the second half does not contain an error.
22 WBEN	Global write burst enable to XBAR slave port designated by FXSBE*n*

Table continues on the next page...

Table 7-16. FBOMCR field description (continued)

Field	Description
	0 Write bursting to all XBAR slave ports is disabled. 1 Write bursting is enabled to any XBAR slave port whose FXSBE*n* bit is asserted.
23 RBEN	Global read burst enable from XBAR slave port designated by FXSBE*n* 0 Read bursting from all XBAR slave ports is disabled. 1 Read bursting is enabled from any XBAR slave port whose FXSBE*n* bit is asserted.
24-31 FXSBE[7:0]	The FEC burst optimization master control register (FBOMCR) controls FEC burst optimization behavior on the system bus. FXSBE — FEC XBAR slave burst enable. FXSBE*n* enables bursting by the FEC interface to the XBAR slave port controlled by that respective FXSBE*n* bit. If FXSBEn is asserted, then that XBAR slave port enabled by the bit can accept the bursts allowed by RBEN and WBEN. Otherwise, the FEC interface will not burst to the XBAR slave port controlled by that respective FXSBE*n* bit. Read bursts from that XBAR slave port are enabled by RBEN. Write bursts to that XBAR slave port are enabled by WBEN. FXSBE*n* assignments to XBAR slave ports: FXSBE0 = Flash FXSBE1 = reserved FXSBE2 = RAM, TCMA, TCMB, TCMC FXSBE3 = reserved FXSBE4 = reserved FXSBE5 = reserved FXSBE6 = reserved FXSBE7 = PBRIDGE0 and PBRIDGE1

7.3.6.2 IAHB Burst Enable 0 (IAHB_BE0) Register

Offset: 04h Access: Supervisor read/write

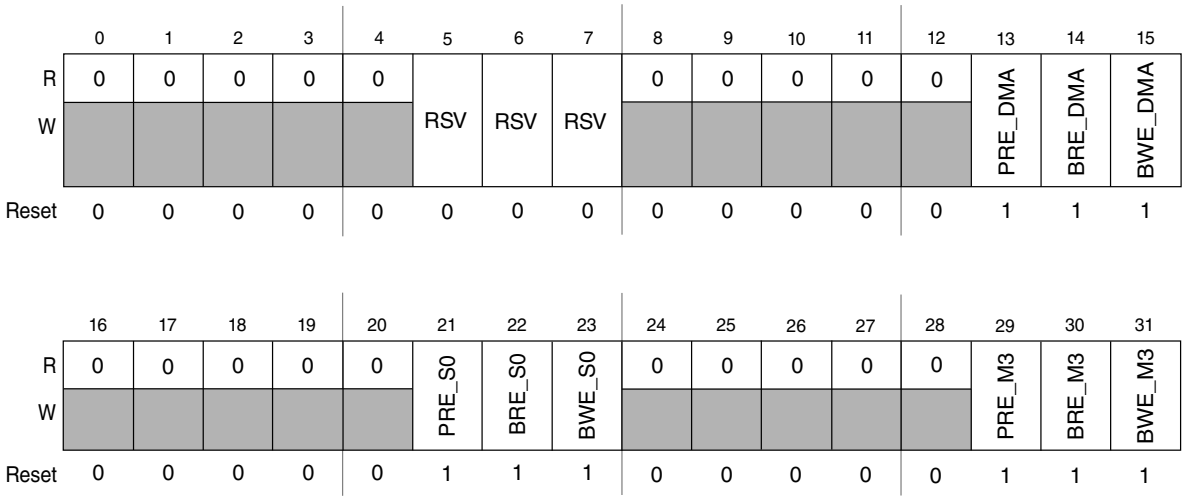


Figure 7-8. IAHB Burst Enable 0 (IAHB_BE0) Register

Table 7-17. IAHB_BE0 Register field description

Field	Description
0–4	Reserved
5–7 RSV	Reserved bits define the PRE, BRE and BWE of the bus gasket for the HSM.
8–12	Reserved
13 PRE_DMA	Pending read enable (PRE) DMA This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_DMA	Burst read enable (BRE) DMA This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_DMA	Burst write enable (BWE) DMA This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20	Reserved
21 PRE_S0	Pending read enable (PRE) S0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_S0	Burst read enable (BRE) S0 This bit controls the bus gasket's handling of burst read transactions.

Table continues on the next page...

Table 7-17. IAHB_BE0 Register field description (continued)

Field	Description	
	0	Burst reads are converted into a series of single transactions on the slave side of the gasket.
	1	Burst reads are optimized for best system performance.
23	Burst write enable (BWE) S0	
BWE_S0	This bit controls the bus gasket's handling of burst write transactions.	
	0	Burst writes are converted into a series of single transactions on the slave side of the gasket.
	1	Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28	Reserved	
29	Pending read enable (PRE) M3	
PRE_M3	This bit controls the bus gasket's handling of pending read transactions.	
	0	Pending reads are disabled.
	1	Pending reads are enabled.
30	Burst read enable (BRE) M3	
BRE_M3	This bit controls the bus gasket's handling of burst read transactions.	
	0	Burst reads are converted into a series of single transactions on the slave side of the gasket.
	1	Burst reads are optimized for best system performance.
31	Burst write enable (BWE) M3	
BWE_M3	This bit controls the bus gasket's handling of burst write transactions.	
	0	Burst writes are converted into a series of single transactions on the slave side of the gasket.
	1	Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.3.6.3 IAHB Burst Enable 1 (IAHB_BE1) Register

Offset: 08h Access: Supervisor read/write

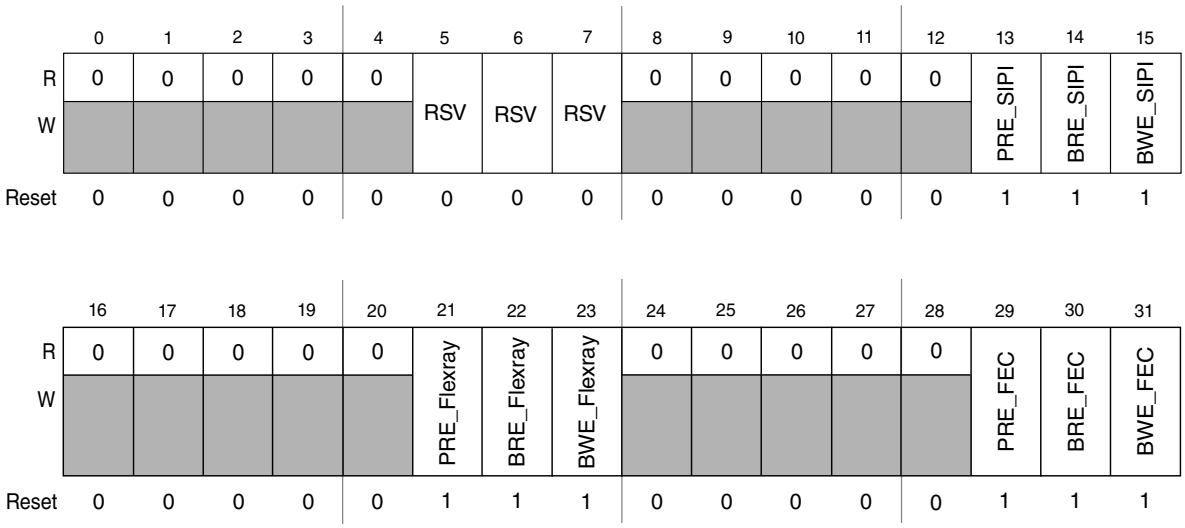


Figure 7-9. IAHB Burst Enable 1 (IAHB_BE1) Register

Table 7-18. IAHB_BE1 field description

Field	Description
0–4	Reserved
5–7 RSV	Reserved, undefined.
8–12	Reserved
13 PRE_SIPI	Pending read enable (PRE) SIPI This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_SIPI	Burst read enable (BRE) SIPI This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_SIPI	Burst write enable (BWE) SIPI This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20	Reserved
21 PRE_Flexray	Pending read enable (PRE) Flexray This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.

Table continues on the next page...

Table 7-18. IAHB_BE1 field description (continued)

Field	Description
22	Burst read enable (BRE) Flexray
BRE_Flexray	This bit controls the bus gasket's handling of burst read transactions.
	0 Burst reads are converted into a series of single transactions on the slave side of the gasket.
	1 Burst reads are optimized for best system performance.
23	Burst write enable (BWE) Flexray
BWE_Flexray	This bit controls the bus gasket's handling of burst write transactions.
	0 Burst writes are converted into a series of single transactions on the slave side of the gasket.
	1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28	Reserved
29	Pending read enable (PRE) FEC
PRE_FEC	This bit controls the bus gasket's handling of pending read transactions.
	0 Pending reads are disabled.
	1 Pending reads are enabled.
30	Burst read enable (BRE) FEC
BRE_FEC	This bit controls the bus gasket's handling of burst read transactions.
	0 Burst reads are converted into a series of single transactions on the slave side of the gasket.
	1 Burst reads are optimized for best system performance.
31	Burst write enable (BWE) FEC
BWE_FEC	This bit controls the bus gasket's handling of burst write transactions.
	0 Burst writes are converted into a series of single transactions on the slave side of the gasket.
	1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.3.6.4 IAHB Burst Enable 2 (IAHB_BE2) Register

Offset: 08h Access: Supervisor read/write

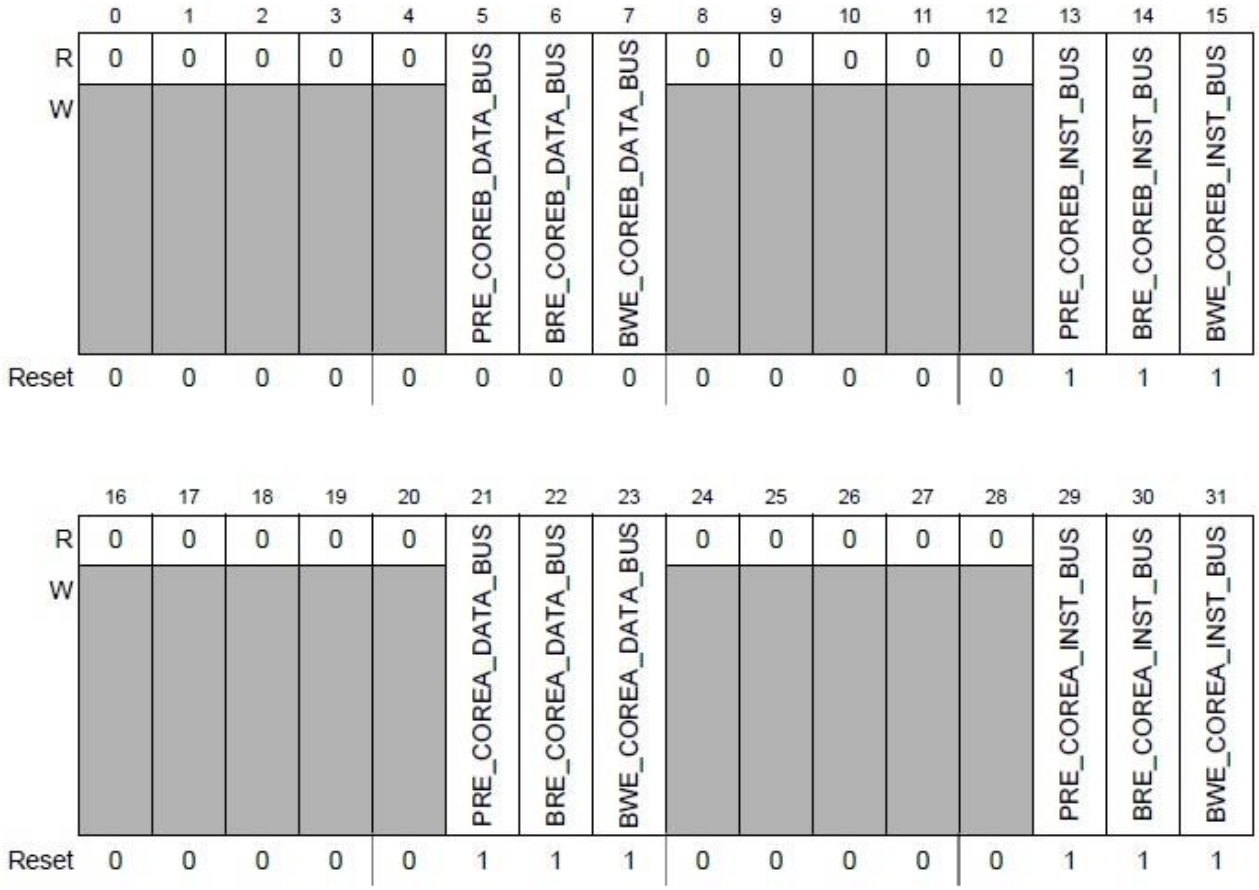


Figure 7-10. IAHB Burst Enable 2 (IAHB_BE2) Register

Table 7-19. IAHB_BE2 field description

Field	Description
0–4	Reserved
5 PRE_COREB_DATA_BUS	Pending read enable (PRE) Core B data bus This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
6 BRE_COREB_DATA_BUS	Burst read enable (BRE) COREB_DATA_BUS This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
7	Burst write enable (BWE) COREB_DATA_BUS

Table continues on the next page...

Table 7-19. IAHB_BE2 field description (continued)

Field	Description
BWE_COREB_DATA_BUS	This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
8-12	Reserved
13 PRE_COREB_INST_BUS	Pending read enable (PRE) Core B instruction Bus This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_COREB_INST_BUS	Burst read enable (BRE) Core B instruction Bus This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_COREB_INST_BUS	Burst write enable (BWE) Core B instruction bus This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16-20	Reserved
21 PRE_COREA_DATA_BUS	Pending read enable (PRE) Core A data bus This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_COREA_DATA_BUS	Burst read enable (BRE) COREA_DATA_BUS This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_COREA_DATA_BUS	Burst write enable (BWE) COREA_DATA_BUS This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket.

Table continues on the next page...

Table 7-19. IAHB_BE2 field description (continued)

Field	Description
	1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24-28	Reserved
29 PRE_COREA_INST_BUS	Pending read enable (PRE) Core A instruction Bus This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_COREA_INST_BUS	Burst read enable (BRE) Core A instruction Bus This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_COREA_INST_BUS	Burst write enable (BWE) Core A instruction bus This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.3.6.5 IAHB Burst Enable 3 (IAHB_BE3) Register

Offset: 0Ch Access: Supervisor read/write

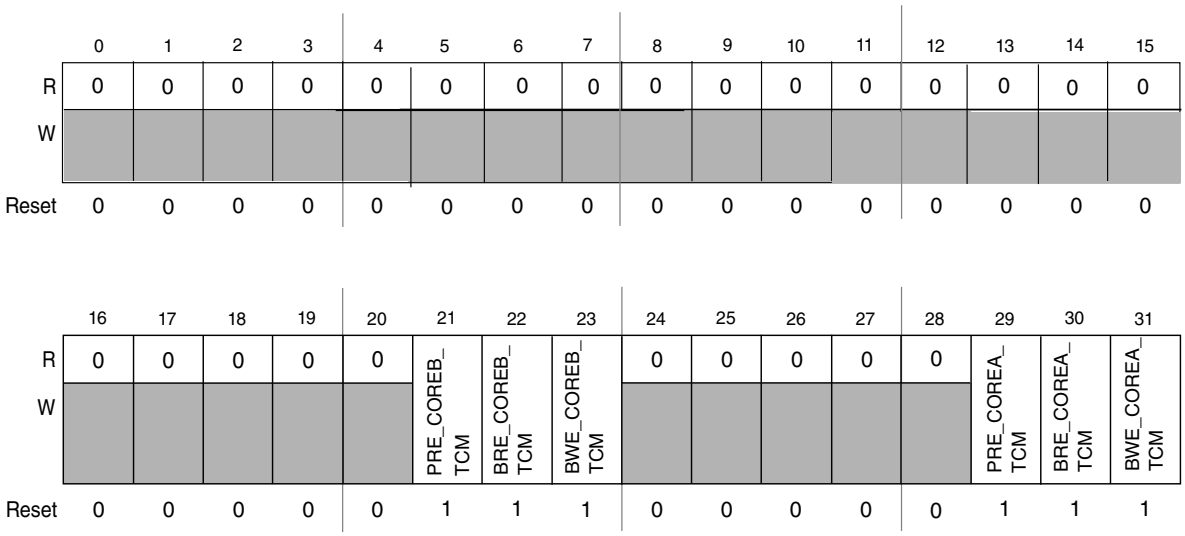


Figure 7-11. IAHB Burst Enable 3 (IAHB_BE3) Register

Table 7-20. IAHB_BE3 field description

Field	Description
0–15	Reserved
16-20	Reserved
21 PRE_COREB_TCM	Pending read enable (PRE) Core B TCM This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_COREB_TCM	Burst read enable (BRE) CORE B TCM This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_COREB_TCM	Burst write enable (BWE) CORE B TCM This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24-28	Reserved
29 PRE_COREA_TCM	Pending read enable (PRE) Core A TCM This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.

Table continues on the next page...

Table 7-20. IAHB_BE3 field description (continued)

Field	Description
30 BRE_COREA_TCM	Burst read enable (BRE) Core A TCM This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_COREA_TCM	Burst write enable (BWE) Core A TCM This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.3.6.6 IPS Clock Gating Module Enable 0 (IPS_CGM_EN0) Register

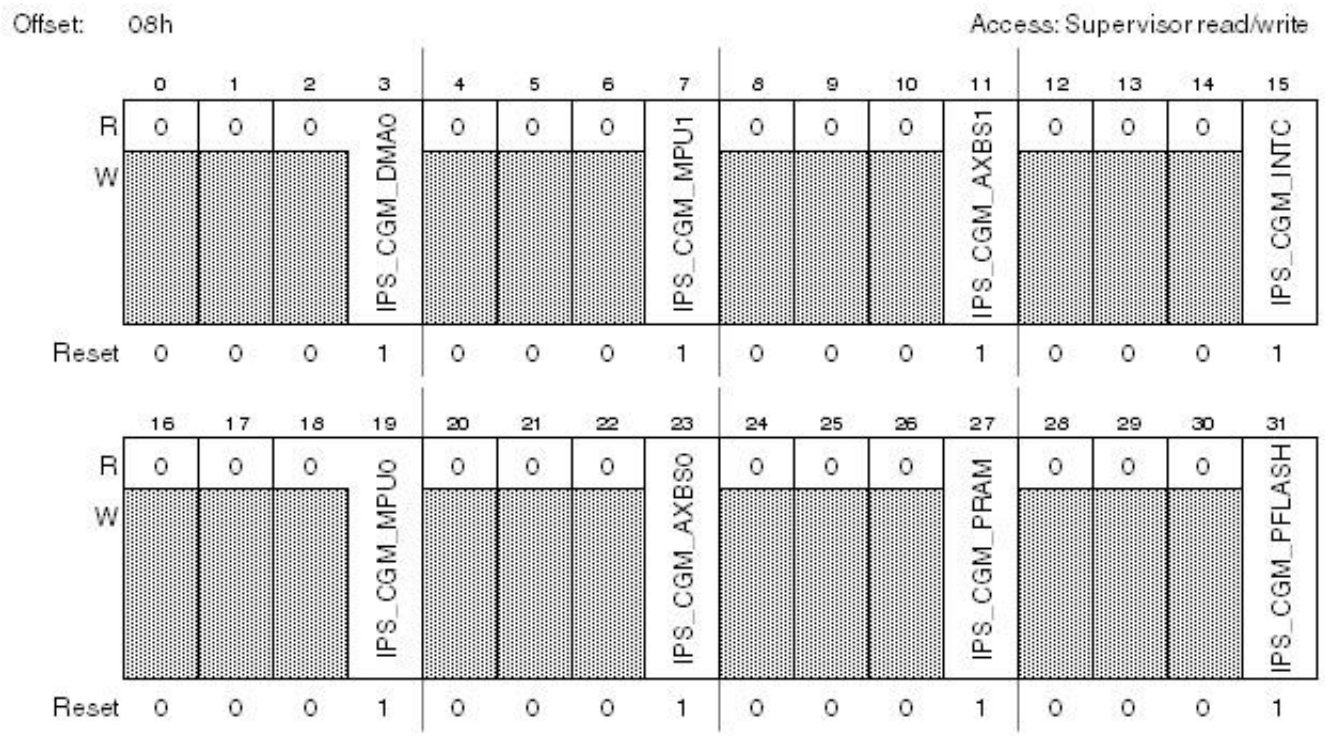


Figure 7-12. IPS Clock Gating Module Enable 0 (IPS_CGM_EN0) Register

Table 7-21. IPS_CGM_EN0 field description

Field	Description
0–2	Reserved

Table continues on the next page...

Table 7-21. IPS_CGM_EN0 field description (continued)

Field	Description
3 IPS_CGM_DMA0	IPS clock gating enable for DMA0 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
4-6	Reserved
7 IPS_CGM_MPU1	IPS clock gating enable for MPU1 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
8-10	Reserved
11 IPS_CGM_AXBS1	IPS clock gating enable for AXBS1 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
12-14	Reserved
15 IPS_CGM_INTC	IPS clock gating enable for INTC 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
16-18	Reserved
19 IPS_CGM_MPU0	IPS clock gating enable for MPU0 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
20-22	Reserved
23 IPS_CGM_AXBS0	IPS clock gating enable for AXBS0 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
24-26	Reserved
27 IPS_CGM_PRAM	IPS clock gating enable for PRAM Controller 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
28-30	Reserved
31 IPS_CGM_PFLASH	IPS clock gating enable for PFlash Controller 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.

7.3.7 Interrupt Controller (INTC) configuration

7.3.7.1 INTC configurable parameters

The INTC on this chip supports 4 processors.

7.3.7.2 INTC implemented registers

Table 7-22 shows the registers implemented on the INTC on this chip.

Table 7-22. INTC implemented registers

Address offset	Register
000h	INTC Block Configuration Register (INTC_BCR)
010h	INTC Current Priority Register for Processor 0 (INTC_CPR0)
014h	INTC Current Priority Register for Processor 1(INTC_CPR1)
018h	INTC Current Priority Register for Processor 2 (INTC_CPR2)
01Ch	INTC Current Priority Register for Processor 3 (INTC_CPR3)
020h	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)
024h	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)
028h	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)
02Ch	INTC Interrupt Acknowledge Register for Processor 3 (INTC_IACKR3)
030h	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)
034h	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)
038h	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)
03Ch	INTC End Of Interrupt Register for Processor 3 (INTC_EOIR3)
040h-05Ch	INTC Software Set/Clear Interrupt Register 0-3 (INTC_SSCIR0_3) - INTC Software Set/Clear Interrupt Register 28-31 (INTC_SSCIR28_31)
05Ch-85Ch	INTC Priority Select Register 0-1 (INTC_PSR0_1) - INTC Priority Select Register 1022-1023 (INTC_PSR1022_1023)

7.3.7.3 Interrupt sources

The following table defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 7-23. Interrupt sources

IRQ #	Offset	Source description	Source name
0		Software settable flag 0	INTC_SSCIR0[CLR0]
1		Software settable flag 1	INTC_SSCIR0[CLR1]
2		Software settable flag 2	INTC_SSCIR0[CLR2]
3		Software settable flag 3	INTC_SSCIR0[CLR3]
4		Software settable flag 4	INTC_SSCIR0[CLR4]
5		Software settable flag 5	INTC_SSCIR0[CLR5]
6		Software settable flag 6	INTC_SSCIR0[CLR6]
7		Software settable flag 7	INTC_SSCIR0[CLR7]
8		Software settable flag 8	INTC_SSCIR0[CLR8]
9		Software settable flag 9	INTC_SSCIR0[CLR9]
10		Software settable flag 10	INTC_SSCIR0[CLR10]
11		Software settable flag 11	INTC_SSCIR0[CLR11]
12		Software settable flag 12	INTC_SSCIR0[CLR12]
13		Software settable flag 13	INTC_SSCIR0[CLR13]
14		Software settable flag 14	INTC_SSCIR0[CLR14]
15		Software settable flag 15	INTC_SSCIR0[CLR15]
16		Software settable flag 16	INTC_SSCIR0[CLR16]
17		Software settable flag 17	INTC_SSCIR0[CLR17]
18		Software settable flag 18	INTC_SSCIR0[CLR18]
19		Software settable flag 19	INTC_SSCIR0[CLR19]
20		Software settable flag 20	INTC_SSCIR0[CLR20]
21		Software settable flag 21	INTC_SSCIR0[CLR21]
22		Software settable flag 22	INTC_SSCIR0[CLR22]
23		Software settable flag 23	INTC_SSCIR0[CLR23]
24		Software settable flag 24	INTC_SSCIR0[CLR24]
25		Software settable flag 25	INTC_SSCIR0[CLR25]
26		Software settable flag 26	INTC_SSCIR0[CLR26]
27		Software settable flag 27	INTC_SSCIR0[CLR27]
28		Software settable flag 28	INTC_SSCIR0[CLR28]
29		Software settable flag 29	INTC_SSCIR0[CLR29]
30		Software settable flag 30	INTC_SSCIR0[CLR30]
31		Software settable flag 31	INTC_SSCIR0[CLR31]
32		Platform watchdog timer0	SWT_0_IR[TIF]
33		Platform watchdog timer1	SWT_1_IR[TIF]
34		Platform watchdog timer2	SWT_2_IR[TIF]
35		Platform watchdog timer3	SWT_3_IR[TIF]
36		Platform periodic timer 0_0 (STM)	STM_0_CIR0[CIF]
37		Platform periodic timer 0_1 (STM)	STM_0_CIR1[CIF]
38		Platform periodic timer 0_2 (STM)	STM_0_CIR2[CIF]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
39		Platform periodic timer 0_3 (STM)	STM_0_CIR3[CIF]
40		Platform periodic timer 1_0 (STM)	STM_1_CIR0[CIF]
41		Platform periodic timer 1_1 (STM)	STM_1_CIR1[CIF]
42		Platform periodic timer 1_2 (STM)	STM_1_CIR2[CIF]
43		Platform periodic timer 1_3 (STM)	STM_1_CIR3[CIF]
44		Platform periodic timer 2_0 (STM)	STM_2_CIR0[CIF]
45		Platform periodic timer 2_1 (STM)	STM_2_CIR1[CIF]
46		Platform periodic timer 2_2 (STM)	STM_2_CIR2[CIF]
47		Platform periodic timer 2_3 (STM)	STM_2_CIR3[CIF]
52		eDMA Combined Error 127 - 0	eDMA Channel Error Flags
53		eDMA Channel 0	DMA_INTL[INT0]
54		eDMA Channel 1	DMA_INTL[INT1]
55		eDMA Channel 2	DMA_INTL[INT2]
56		eDMA Channel 3	DMA_INTL[INT3]
57		eDMA Channel 4	DMA_INTL[INT4]
58		eDMA Channel 5	DMA_INTL[INT5]
59		eDMA Channel 6	DMA_INTL[INT6]
60		eDMA Channel 7	DMA_INTL[INT7]
61		eDMA Channel 8	DMA_INTL[INT8]
62		eDMA Channel 9	DMA_INTL[INT9]
63		eDMA Channel 10	DMA_INTL[INT10]
64		eDMA Channel 11	DMA_INTL[INT11]
65		eDMA Channel 12	DMA_INTL[INT12]
66		eDMA Channel 13	DMA_INTL[INT13]
67		eDMA Channel 14	DMA_INTL[INT14]
68		eDMA Channel 15	DMA_INTL[INT15]
69		eDMA Channel 16	DMA_INTL[INT16]
70		eDMA Channel 17	DMA_INTL[INT17]
71		eDMA Channel 18	DMA_INTL[INT18]
72		eDMA Channel 19	DMA_INTL[INT19]
73		eDMA Channel 20	DMA_INTL[INT20]
74		eDMA Channel 21	DMA_INTL[INT21]
75		eDMA Channel 22	DMA_INTL[INT22]
76		eDMA Channel 23	DMA_INTL[INT23]
77		eDMA Channel 24	DMA_INTL[INT24]
78		eDMA Channel 25	DMA_INTL[INT25]
79		eDMA Channel 26	DMA_INTL[INT26]
80		eDMA Channel 27	DMA_INTL[INT27]
81		eDMA Channel 28	DMA_INTL[INT28]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
82		eDMA Channel 29	DMA_INTL[INT29]
83		eDMA Channel 30	DMA_INTL[INT30]
84		eDMA Channel 31	DMA_INTL[INT31]
85		eDMA Channel 32	DMA_INTH[INT32]
86		eDMA Channel 33	DMA_INTH[INT33]
87		eDMA Channel 34	DMA_INTH[INT34]
88		eDMA Channel 35	DMA_INTH[INT35]
89		eDMA Channel 36	DMA_INTH[INT36]
90		eDMA Channel 37	DMA_INTH[INT37]
91		eDMA Channel 38	DMA_INTH[INT38]
92		eDMA Channel 39	DMA_INTH[INT39]
93		eDMA Channel 40	DMA_INTH[INT40]
94		eDMA Channel 41	DMA_INTH[INT41]
95		eDMA Channel 42	DMA_INTH[INT42]
96		eDMA Channel 43	DMA_INTH[INT43]
97		eDMA Channel 44	DMA_INTH[INT44]
98		eDMA Channel 45	DMA_INTH[INT45]
99		eDMA Channel 46	DMA_INTH[INT46]
100		eDMA Channel 47	DMA_INTH[INT47]
101		eDMA Channel 48	DMA_INTH[INT48]
102		eDMA Channel 49	DMA_INTH[INT49]
103		eDMA Channel 50	DMA_INTH[INT50]
104		eDMA Channel 51	DMA_INTH[INT51]
105		eDMA Channel 52	DMA_INTH[INT52]
106		eDMA Channel 53	DMA_INTH[INT53]
107		eDMA Channel 54	DMA_INTH[INT54]
108		eDMA Channel 55	DMA_INTH[INT55]
109		eDMA Channel 56	DMA_INTH[INT56]
110		eDMA Channel 57	DMA_INTH[INT57]
111		eDMA Channel 58	DMA_INTH[INT58]
112		eDMA Channel 59	DMA_INTH[INT59]
113		eDMA Channel 60	DMA_INTH[INT60]
114		eDMA Channel 61	DMA_INTH[INT61]
115		eDMA Channel 62	DMA_INTH[INT62]
116		eDMA Channel 63	DMA_INTH[INT63]
117		eDMA Channel 64	DMA_INTH[INT64]
118		eDMA Channel 65	DMA_INTH[INT65]
119		eDMA Channel 66	DMA_INTH[INT66]
120		eDMA Channel 67	DMA_INTH[INT67]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
121		eDMA Channel 68	DMA_INTH[INT68]
122		eDMA Channel 69	DMA_INTH[INT69]
123		eDMA Channel 70	DMA_INTH[INT70]
124		eDMA Channel 71	DMA_INTH[INT71]
125		eDMA Channel 72	DMA_INTH[INT72]
126		eDMA Channel 73	DMA_INTH[INT73]
127		eDMA Channel 74	DMA_INTH[INT74]
128		eDMA Channel 75	DMA_INTH[INT75]
129		eDMA Channel 76	DMA_INTH[INT76]
130		eDMA Channel 77	DMA_INTH[INT77]
131		eDMA Channel 78	DMA_INTH[INT78]
132		eDMA Channel 79	DMA_INTH[INT79]
133		eDMA Channel 80	DMA_INTH[INT80]
134		eDMA Channel 81	DMA_INTH[INT81]
135		eDMA Channel 82	DMA_INTH[INT82]
136		eDMA Channel 83	DMA_INTH[INT83]
137		eDMA Channel 84	DMA_INTH[INT84]
138		eDMA Channel 85	DMA_INTH[INT85]
139		eDMA Channel 86	DMA_INTH[INT86]
140		eDMA Channel 87	DMA_INTH[INT87]
141		eDMA Channel 88	DMA_INTH[INT88]
142		eDMA Channel 89	DMA_INTH[INT89]
143		eDMA Channel 90	DMA_INTH[INT90]
144		eDMA Channel 91	DMA_INTH[INT91]
145		eDMA Channel 92	DMA_INTH[INT92]
146		eDMA Channel 93	DMA_INTH[INT93]
147		eDMA Channel 94	DMA_INTH[INT94]
148		eDMA Channel 95	DMA_INTH[INT95]
149		eDMA Channel 96	DMA_INTH[INT96]
150		eDMA Channel 97	DMA_INTH[INT97]
151		eDMA Channel 98	DMA_INTH[INT98]
152		eDMA Channel 99	DMA_INTH[INT99]
153		eDMA Channel 100	DMA_INTH[INT100]
154		eDMA Channel 101	DMA_INTH[INT101]
155		eDMA Channel 102	DMA_INTH[INT102]
156		eDMA Channel 103	DMA_INTH[INT103]
157		eDMA Channel 104	DMA_INTH[INT104]
158		eDMA Channel 105	DMA_INTH[INT105]
159		eDMA Channel 106	DMA_INTH[INT106]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
160		eDMA Channel 107	DMA_INTH[INT107]
161		eDMA Channel 108	DMA_INTH[INT108]
162		eDMA Channel 109	DMA_INTH[INT109]
163		eDMA Channel 110	DMA_INTH[INT110]
164		eDMA Channel 111	DMA_INTH[INT111]
165		eDMA Channel 112	DMA_INTH[INT112]
166		eDMA Channel 113	DMA_INTH[INT113]
167		eDMA Channel 114	DMA_INTH[INT114]
168		eDMA Channel 115	DMA_INTH[INT115]
169		eDMA Channel 116	DMA_INTH[INT116]
170		eDMA Channel 117	DMA_INTH[INT117]
171		eDMA Channel 118	DMA_INTH[INT118]
172		eDMA Channel 119	DMA_INTH[INT119]
173		eDMA Channel 120	DMA_INTH[INT120]
174		eDMA Channel 121	DMA_INTH[INT121]
175		eDMA Channel 122	DMA_INTH[INT122]
176		eDMA Channel 123	DMA_INTH[INT123]
177		eDMA Channel 124	DMA_INTH[INT124]
178		eDMA Channel 125	DMA_INTH[INT125]
179		eDMA Channel 126	DMA_INTH[INT126]
180		eDMA Channel 127	DMA_INTH[INT127]
185		Flash controller Prog/Erase/Suspend IRQ_0	MCR[DONE]
218		Ethernet_0_0	EIR[TXF]
219		Ethernet_0_1	EIR[RXF]
220		Ethernet_0_2	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]
226		Periodic Interrupt Timer (PIT0)	PIT_0_TFLG0[TIF]
227		Periodic Interrupt Timer (PIT1)	PIT_0_TFLG1[TIF]
228		Periodic Interrupt Timer (PIT2)	PIT_0_TFLG2[TIF]
229		Periodic Interrupt Timer (PIT3)	PIT_0_TFLG3[TIF]
230		Periodic Interrupt Timer (PIT4)	PIT_0_TFLG4[TIF]
231		Periodic Interrupt Timer (PIT5)	PIT_0_TFLG5[TIF]
232		Periodic Interrupt Timer (PIT6)	PIT_0_TFLG6[TIF]
233		Periodic Interrupt Timer (PIT7)	PIT_0_TFLG7[TIF]
239		PIT_RTI	PIT_0_RTI_TFLG[TIF]
240		PIT_64_Upper	PIT_1_TFLG0[TIF]
241		PIT_64_Lower	PIT_1_TFLG1[TIF]
242		XOSC counter	XOSC
243		SIU External Interrupt_0	SIU External Interrupt_0

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
244		SIU External Interrupt_1	SIU External Interrupt_1
251		MC_ME 0	ME_IS[I_SAFE]
252		MC_ME 1	ME_IS[I_MTC]
253		MC_ME 2	ME_IS[I_IMODE]
254		MC_ME 3	ME_IS[I_ICONF] | ME_IS[I_ICONF_CC] | ME_IS[I_ICONF_CU
255		MC_RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
259		DSPI0_0	DSPI_0_SR[TFUF] | DSPI_0_SR[RFOF]| DSPI_0_SR[TFIWF]
260		DSPI0_1	DSPI_0_SR[EOQF]
261		DSPI0_2	DSPI_0_SR[TFFF]
262		DSPI0_3	DSPI_0_SR[TCF]
263		DSPI0_4	DDSPI_0_SR[RFDF]
264		DSPI0_5	DSPI_0_SR[CMD_TCF]
265		DSPI0_6	DSPI_0_SR[CMDFFF]
266		DSPI0_7	DSPI_0_SR[SPEF]
268		DSPI1_0	DSPI_1_SR[TFUF] | DSPI_1_SR[RFOF]| DSPI_1_SR[TFIWF]
269		DSPI1_1	DSPI_1_SR[EOQF]
270		DSPI1_2	DSPI_1_SR[TFFF]
271		DSPI1_3	DSPI_1_SR[TCF]
272		DSPI1_4	DSPI_1_SR[RFDF]
273		DSPI1_5	DSPI_1_SR[CMD_TCF]
274		DSPI1_6	DSPI_1_SR[CMDFFF]
275		DSPI1_7	DSPI_1_SR[SPEF]
277		DSPI2_0	DSPI_2_SR[TFUF] | DSPI_2_SR[RFOF]| DSPI_2_SR[TFIWF]
278		DSPI2_1	DSPI_2_SR[EOQF]
279		DSPI2_2	DSPI_2_SR[TFFF]
280		DSPI2_3	DSPI_2_SR[TCF]
281		DSPI2_4	DSPI_2_SR[RFDF]
282		DSPI2_5	DSPI_2_SR[CMD_TCF]
283		DSPI2_6	DSPI_2_SR[CMDFFF]
284		DSPI2_7	DSPI_2_SR[SPEF]
286		DSPI3_0	DSPI_3_SR[TFUF] | DSPI_3_SR[RFOF]| DSPI_3_SR[TFIWF]
287		DSPI3_1	DSPI_3_SR[EOQF]
288		DSPI3_2	DSPI_3_SR[TFFF]
289		DSPI3_3	DSPI_3_SR[TCF]
290		DSPI3_4	DSPI_3_SR[RFDF]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
291		DSPI3_5	DSPI_3_SR[CMD_TCF]
292		DSPI3_6	DSPI_3_SR[CMDFFF]
293		DSPI3_7	DSPI_3_SR[SPEF]
295		DSPI4_0	DSPI_4_SR[TFUF] | DSPI_4_SR[RFOF]| DSPI_4_SR[TFIWF]
296		DSPI4_1	DSPI_4_SR[EOQF]
297		DSPI4_2	DSPI_4_SR[TFFF]
298		DSPI4_3	DSPI_4_SR[TCF]
299		DSPI4_4	DSPI_4_SR[RFDF]
300		DSPI4_5	DSPI_4_SR[SPITCF] | DSPI_4_SR[CMD_TCF]
301		DSPI4_6	DSPI_4_SR[DSITCF] | DSPI_4_SR[CMDFFF]
302		DSPI4_7	DSPI_4_SR[SPEF] | DSPI_4_SR[DPEF]
303		DSPI4_8	DSPI_4_SR[DDIF]
304		DSPI5_0	DSPI_5_SR[TFUF] | DSPI_5_SR[RFOF]| DSPI_5_SR[TFIWF]
305		DSPI5_1	DSPI_5_SR[EOQF]
306		DSPI5_2	DSPI_5_SR[TFFF]
307		DSPI5_3	DSPI_5_SR[TCF]
308		DSPI5_4	DSPI_5_SR[RFDF]
309		DSPI5_5	DSPI_5_SR[SPITCF] | DSPI_5_SR[CMD_TCF]
310		DSPI5_6	DSPI_5_SR[DSITCF] | DSPI_5_SR[CMDFFF]
311		DSPI5_7	DSPI_5_SR[SPEF] | DSPI_5_SR[DPEF]
312		DSPI5_8	DSPI_5_SR[DDIF]
313		DSPI6_0	DSPI_6_SR[TFUF] | DSPI_6_SR[RFOF]| DSPI_6_SR[TFIWF]
314		DSPI6_1	DSPI_6_SR[EOQF]
315		DSPI6_2	DSPI_6_SR[TFFF]
316		DSPI6_3	DSPI_6_SR[TCF]
317		DSPI6_4	DSPI_6_SR[RFDF]
318		DSPI6_5	DSPI_6_SR[SPITCF] | DSPI_6_SR[CMD_TCF]
319		DSPI6_6	DSPI_6_SR[DSITCF] | DSPI_6_SR[CMDFFF]
320		DSPI6_7	DSPI_6_SR[SPEF] | DSPI_6_SR[DPEF]
321		DSPI6_8	DSPI_6_SR[DDIF]
367		DSPI12_0	DSPI_12_SR[TFUF] | DSPI_12_SR[RFOF]| DSPI_12_SR[TFIWF]
368		DSPI12_1	DSPI_12_SR[EOQF]
369		DSPI12_2	DSPI_12_SR[TFFF]
370		DSPI12_3	DSPI_12_SR[TCF]
371		DSPI12_4	DSPI_12_SR[RFDF]
372		DSPI_12_5	DSPI_12_SR[CMD_TCF]
373		DSPI_12_6	DSPI_12_SR[CMDFFF]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
374		DSPI12_7	DSPI_12_SR[SPEF]
376		Linflex/eSCI0_0	LINFlex_0_RXI
377		Linflex/eSCI0_1	LINFlex_0_TXI
378		Linflex/eSCI0_2	LINFlex_0_ERR
380		Linflex/eSCI1_0	LINFlex_1_RXI
381		Linflex/eSCI1_1	LINFlex_1_TXI
382		Linflex/eSCI1_2	LINFlex_1_ERR
384		Linflex/eSCI2_0	LINFlex_2_RXI
385		Linflex/eSCI2_1	LINFlex_2_TXI
386		Linflex/eSCI2_2	LINFlex_2_ERR
416		Linflex/eSCI16_0	LINFlex_16_RXI
417		Linflex/eSCI16_1	LINFlex_16_TXI
418		Linflex/eSCI16_2	LINFlex_16_ERR
432		Linflex/eSCI14_0	LINFlex_14_RXI
433		Linflex/eSCI14_1	LINFlex_14_TXI
434		Linflex/eSCI14_2	LINFlex_14_ERR
436		Linflex/eSCI15_0	LINFlex_15_RXI
437		Linflex/eSCI15_1	LINFlex_15_TXI
438		Linflex/eSCI15_2	LINFlex_15_ERR
440		IIC_0_0	I2C0_SR[IBAL] | I2C0_SR[TCF] | I2C0_SR[IAAS]
442		IIC_1_0	I2C1_SR[IBAL] | I2C1_SR[TCF] | I2C1_SR[IAAS]
453		FlexRay_0_0	FR_0_LRNEIF | DRNEIF
454		FlexRay_0_1	FR_0_LRCEIF | DRCEIF
455		FlexRay_0_2	FR_0_FNEAIF
456		FlexRay_0_3	FR_0_FNEBIF
457		FlexRay_0_4	FR_0_WUPIF
458		FlexRay_0_5	FR_0_PRIF
459		FlexRay_0_6	FR_0_CHIF
460		FlexRay_0_7	FR_0_TBIF
461		FlexRay_0_8	FR_0_RBIF
462		FlexRay_0_9	FR_0_MIF
465		FlexRay_1_0	FR_1_LRNEIF | DRNEIF
466		FlexRay_1_1	FR_1_LRCEIF | DRCEIF
467		FlexRay_1_2	FR_1_FNEAIF
468		FlexRay_1_3	FR_1_FNEBIF
469		FlexRay_1_4	FR_1_WUPIF
470		FlexRay_1_5	FR_1_PRIF
471		FlexRay_1_6	FR_1_CHIF

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
472		FlexRay_1_7	FR_1_TBIF
473		FlexRay_1_8	FR_1_RBIF
474		FlexRay_1_9	FR_1_MIF
477		Power Monitor Unit	GR_S[VD15] | GR_S[VD14] | GR_S[VD13] | GR_S[VD12] | GR_S[VD10] | GR_S[VD9] | GR_S[VD7] | GR_S[VD4] | GR_S[VD3]
478		Power management Unit (temp sensor)	EPR_TD[TEMP_3] | EPR_TD[TEMP_2] | EPR_TD[TEMP_0]
488		FCCU_0	FCCU_IRQ_STAT[ALRM_STAT]
489		FCCU_1	FCU_IRQ_STAT[CFG_TO_STAT]
494		STCU2_0	STCU_RUNSW[LBIE]
495		STCU2_1	STCU_RUNSW[MBIE]
496		Hardware Security Module 0	HSM2HTIE[0]
497		Hardware Security Module 1	HSM2HTIE[1]
498		Hardware Security Module 2	HSM2HTIE[2]
499		Hardware Security Module 3	HSM2HTIE[3]
500		Hardware Security Module 4	HSM2HTIE[4]
501		Hardware Security Module 5	HSM2HTIE[5]
502		Hardware Security Module 6	HSM2HTIE[6]
503		Hardware Security Module 7	HSM2HTIE[7]
504		Hardware Security Module 8	HSM2HTIE[8]
505		Hardware Security Module 9	HSM2HTIE[9]
506		Hardware Security Module 10	HSM2HTIE[10]
507		Hardware Security Module 11	HSM2HTIE[11]
508		Hardware Security Module 12	HSM2HTIE[12]
509		Hardware Security Module 13	HSM2HTIE[13]
510		Hardware Security Module 14	HSM2HTIE[14]
511		Hardware Security Module 15	HSM2HTIE[15]
512		Hardware Security Module 16	HSM2HTIE[16]
513		Hardware Security Module 17	HSM2HTIE[17]
514		Hardware Security Module 18	HSM2HTIE[18]
515		Hardware Security Module 19	HSM2HTIE[19]
516		Hardware Security Module 20	HSM2HTIE[20]
517		Hardware Security Module 21	HSM2HTIE[21]
518		Hardware Security Module 22	HSM2HTIE[22]
519		Hardware Security Module 23	HSM2HTIE[23]
520		Hardware Security Module 24	HSM2HTIE[24]
521		Hardware Security Module 25	HSM2HTIE[25]
522		Hardware Security Module 26	HSM2HTIE[26]
523		Hardware Security Module 27	HSM2HTIE[27]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
524		Hardware Security Module 28	HSM2HTIE[28]
525		Hardware Security Module 29	HSM2HTIE[29]
526		Hardware Security Module 30	HSM2HTIE[30]
527		Hardware Security Module 31	HSM2HTIE[31]
528		SARADC_0	SARADC_0
529		SARADC_1	SARADC_1
530		SARADC_2	SARADC_2
531		SARADC_3	SARADC_3
532		SARADC_4	SARADC_4
533		SARADC_5	Reserved for SARADC_5
534		SARADC_6	SARADC_6
535		SARADC_7	SARADC_7
536		SARADC_8	SARADC_8
537		SARADC_9	SARADC_9
538		SARADC_10	SARADC_10
543		SARADC_B	SARADC_B
544		SDADC_0	SDADC_0
545		SDADC_1	SDADC_1
546		SDADC_2	SDADC_2
547		SDADC_3	SDADC_3
548		SDADC_4	SDADC_4
549		SDADC_5	SDADC_5
550		SDADC_6	SDADC_6
551		SDADC_7	SDADC_7
552		SDADC_8	SDADC_8
553		SDADC_9	SDADC_9
558		SENT_COMBINED_FAST_0	SENT_0_FMSG_RDY[0] | SENT_0_FMSG_RDY[1] | SENT_0_FMSG_RDY[2] | SENT_0_FMSG_RDY[3] | SENT_0_FMSG_RDY[4] | SENT_0_FMSG_RDY[5] | SENT_0_FMSG_RDY[6] | SENT_0_FMSG_RDY[7]
559		SENT_COMBINED_SLOW_0	SENT_0_SMSG_RDY[0] | SENT_0_SMSG_RDY[1] | SENT_0_SMSG_RDY[2] | SENT_0_SMSG_RDY[3] | SENT_0_SMSG_RDY[4] | SENT_0_SMSG_RDY[5] | SENT_0_SMSG_RDY[6] | SENT_0_SMSG_RDY[7]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
560		SENT_COMBINED_ERR_0	SENT_0_GBL_STATUS[FMDU] | SENT_0_GBL_STATUS[SMDU] | SENT_0_CHn_STATUS[4:7] | SENT_0_CHn_STATUS[9:15]
561		SENT_COMBINED_FAST_1	SENT_1_FMSG_RDY[0] | SENT_1_FMSG_RDY[1] | SENT_1_FMSG_RDY[2] | SENT_1_FMSG_RDY[3] | SENT_1_FMSG_RDY[4] | SENT_1_FMSG_RDY[5] | SENT_1_FMSG_RDY[6]
562		SENT_COMBINED_SLOW_1	SENT_1_SMSG_RDY[0] | SENT_1_SMSG_RDY[1] | SENT_1_SMSG_RDY[2] | SENT_1_SMSG_RDY[3] | SENT_1_SMSG_RDY[4] | SENT_1_SMSG_RDY[5] | SENT_1_SMSG_RDY[6]
563		SENT_COMBINED_ERR_1	SENT_1_GBL_STATUS[FMDU] | SENT_1_GBL_STATUS[SMDU] | SENT_1_CHn_STATUS[4:7] | SENT_1_CHn_STATUS[9:15]
564		SENT_0_CH0_FAST	SENT_0_FMSG_RDY[0]
565		SENT_0_CH0_SLOW	SENT_0_SMSG_RDY[0]
566		SENT_0_CH0_ERR	SENT_0_CH0_STATUS[4:7] | SENT_0_CH0_STATUS[9:15]
567		SENT_0_CH1_FAST	SENT_0_FMSG_RDY[1]
568		SENT_0_CH1_SLOW	SENT_0_SMSG_RDY[1]
569		SENT_0_CH1_ERR	SENT_0_CH1_STATUS[4:7] | SENT_0_CH1_STATUS[9:15]
570		SENT_0_CH2_FAST	SENT_0_FMSG_RDY[2]
571		SENT_0_CH2_SLOW	SENT_0_SMSG_RDY[2]
572		SENT_0_CH2_ERR	SENT_0_CH2_STATUS[4:7] | SENT_2_CH3_STATUS[9:15]
573		SENT_0_CH3_FAST	SENT_0_FMSG_RDY[3]
574		SENT_0_CH3_SLOW	SENT_0_SMSG_RDY[3]
575		SENT_0_CH3_ERR	SENT_0_CH3_STATUS[4:7] | SENT_0_CH3_STATUS[9:15]
576		SENT_0_CH4_FAST	SENT_0_FMSG_RDY[4]
577		SENT_0_CH4_SLOW	SENT_0_SMSG_RDY[4]
578		SENT_0_CH4_ERR	SENT_0_CH4_STATUS[4:7] | SENT_0_CH4_STATUS[9:15]
579		SENT_1_CH0_FAST	SENT_1_FMSG_RDY[0]
580		SENT_1_CH0_SLOW	SENT_1_SMSG_RDY[0]
581		SENT_1_CH0_ERR	SENT_1_CH0_STATUS[4:7] | SENT_1_CH0_STATUS[9:15]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
582		SENT_1_CH1_FAST	SENT_1_FMSG_RDY[1]
583		SENT_1_CH1_SLOW	SENT_1_SMSG_RDY[1]
584		SENT_1_CH1_ERR	SENT_1_CH1_STATUS[4:7] | SENT_1_CH1_STATUS[9:15]
585		SENT_1_CH2_FAST	SENT_1_FMSG_RDY[2]
586		SENT_1_CH2_SLOW	SENT_1_SMSG_RDY[2]
587		SENT_1_CH2_ERR	SENT_1_CH3_STATUS[4:7] | SENT_1_CH3_STATUS[9:15]
588		SENT_1_CH3_FAST	SENT_1_FMSG_RDY[3]
589		SENT_1_CH3_SLOW	SENT_1_SMSG_RDY[3]
590		SENT_1_CH3_ERR	SENT_1_CH3_STATUS[4:7] | SENT_1_CH3_STATUS[9:15]
591		SENT_1_CH4_FAST	SENT_1_FMSG_RDY[4]
592		SENT_1_CH4_SLOW	SENT_1_SMSG_RDY[4]
593		SENT_1_CH4_ERR	SENT_1_CH4_STATUS[4:7] | SENT_1_CH4_STATUS[9:15]
594		SENT_0_CH5_FAST	SENT_0_FMSG_RDY[5]
595		SENT_0_CH5_SLOW	SENT_0_SMSG_RDY[5]
596		SENT_0_CH5_ERR	SENT_0_CH5_STATUS[4:7] | SENT_0_CH5_STATUS[9:15]
597		SENT_0_CH6_FAST	SENT_0_FMSG_RDY[6]
598		SENT_0_CH6_SLOW	SENT_0_SMSG_RDY[6]
599		SENT_0_CH6_ERR	SENT_0_CH6_STATUS[4:7] | SENT_0_CH6_STATUS[9:15]
600		SENT_0_CH7_FAST	SENT_0_FMSG_RDY[7]
601		SENT_0_CH7_SLOW	SENT_0_SMSG_RDY[7]
602		SENT_0_CH7_ERR	SENT_0_CH7_STATUS[4:7] | SENT_0_CH7_STATUS[9:15]
603		SENT_1_CH5_FAST	SENT_1_FMSG_RDY[5]
604		SENT_1_CH5_SLOW	SENT_1_SMSG_RDY[5]
605		SENT_1_CH5_ERR	SENT_1_CH5_STATUS[4:7] | SENT_1_CH5_STATUS[9:15]
606		SENT_1_CH6_FAST	SENT_1_FMSG_RDY[6]
607		SENT_1_CH6_SLOW	SENT_1_SMSG_RDY[6]
608		SENT_1_CH6_ERR	SENT_1_CH6_STATUS[4:7] | SENT_1_CH6_STATUS[9:15]
612		PSI5_0_CH0_0	PSI5_0_CH_0_DSR[IS_DMA_TF_PM_DS] | PSI5_0_CH_0_DSR[IS_DMA_TF_SF] | PSI5_0_CH_0_DSR[IS_DMA_PM_DS_FIFO_FULL] | PSI5_0_CH_0_DSR[IS_DMA_SFUF] | PSI5_0_CH_0_DSR[IS_DMA_PM_DS_UF]
613		PSI5_0_CH0_1	PSI5_0_CH_0_GISR[IS_CESM[6:1]] | PSI5_0_CH_0_GISR[IS_STS] | PSI5_0_CH_0_GISR[IS_DTS] |

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
			PSI5_0_CH_0_GISR[[IS_DSROW] | PSI5_0_CH_0_GISR[IS_BROW] | PSI5_0_CH_0_GISR[[DSR_RDY] | PSI5_0_CH_0_GISR[DBR_RDY] | PSI5_0_CH_0_GISR[DPR_RDY] | PSI5_0_CH_0_GISR[IS_OWSM[6:1]] | PSI5_0_CH_0_GISR[IS_NVSM[6:1]]
614		PSI5_0_CH0_2	PSI5_0_CH_0_NDSR[NDS31] | PSI5_0_CH_0_NDSR[NDS30] | … | PSI5_0_CH_0_NDSR[NDS1] | PSI5_0_CH_0_NDSR[NDS0]
615		PSI5_0_CH0_3	PSI5_0_CH_0_OWSR[NDS31] | PSI5_0_CH_0_OWSR[NDS30] | … | PSI5_0_CH_0_OWSR[NDS1] | PSI5_0_CH_0_OWSR[NDS0]
616		PSI5_0_CH0_4	PSI5_0_CH_0_EISR[NDS31] | PSI5_0_CH_0_EISR[NDS30] | … | PSI5_0_CH_0_EISR[NDS1] | PSI5_0_CH_0_EISR[NDS0]
617		PSI5_0_CH0_5	PSI5_0_CH_0_DSR[*] | PSI5_0_CH_0_GISR[*] | PSI5_0_CH_0_NDSR[31:0] | PSI5_0_CH_0_OWSR[31:0] | PSI5_0_CH_0_EISR[31:0]
618		PSI5_0_CH1_0	PSI5_0_CH_1_DSR[IS_DMA_TF_PM_DS] | PSI5_0_CH_1_DSR[IS_DMA_TF_SF] | PSI5_0_CH_1_DSR[IS_DMA_PM_DS_FIFO_FULL] | PSI5_0_CH_1_DSR[IS_DMA_SFUF] | PSI5_0_CH_1_DSR[IS_DMA_PM_DS_UF]
619		PSI5_0_CH1_1	PSI5_0_CH_1_GISR[IS_CESM[6:1]] | PSI5_0_CH_1_GISR[IS_STS] | PSI5_0_CH_1_GISR[IS_DTS] | PSI5_0_CH_1_GISR[[IS_DSROW] | PSI5_0_CH_1_GISR[IS_BROW] | PSI5_0_CH_1_GISR[[DSR_RDY] | PSI5_0_CH_1_GISR[DBR_RDY] | PSI5_0_CH_1_GISR[DPR_RDY] | PSI5_0_CH_1_GISR[IS_OWSM[6:1]] | PSI5_0_CH_1_GISR[IS_NVSM[6:1]]
620		PSI5_0_CH1_2	PSI5_0_CH_1_NDSR[NDS31] | PSI5_0_CH_1_NDSR[NDS30] | Â… | PSI5_0_CH_1_NDSR[NDS1] | PSI5_0_CH_1_NDSR[NDS0]
621		PSI5_0_CH1_3	PSI5_0_CH_1_OWSR[NDS31] | PSI5_0_CH_1_OWSR[NDS30] | … | PSI5_0_CH_1_OWSR[NDS1] | PSI5_0_CH_1_OWSR[NDS0]
622		PSI5_0_CH1_4	PSI5_0_CH_1_EISR[NDS31] | PSI5_0_CH_1_EISR[NDS30] | … | PSI5_0_CH_1_EISR[NDS1] | PSI5_0_CH_1_EISR[NDS0]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
623		PSI5_0_CH1_5	PSI5_0_CH_1_DSR[*] | PSI5_0_CH_1_GISR[*] | PSI5_0_CH_1_NDSR[31:0] | PSI5_0_CH_1_OWSR[31:0] | PSI5_0_CH_1_EISR[31:0]
624		PSI5_1_CH0_0	PSI5_1_CH_0_DSR[IS_DMA_TF_PM_DS] | PSI5_1_CH_0_DSR[IS_DMA_TF_SF] | PSI5_1_CH_0_DSR[IS_DMA_PM_DS_FIFO_FULL] | PSI5_1_CH_0_DSR[IS_DMA_SFUF] | PSI5_1_CH_0_DSR[IS_DMA_PM_DS_UF]
625		PSI5_1_CH0_1	PSI5_1_CH_0_GISR[IS_CESM[6:1]] | PSI5_1_CH_0_GISR[IS_STS] | PSI5_1_CH_0_GISR[IS_DTS] | PSI5_1_CH_0_GISR[[IS_DSROW] | PSI5_1_CH_0_GISR[IS_BROW] | PSI5_1_CH_0_GISR[[DSR_RDY] | PSI5_1_CH_0_GISR[DBR_RDY] | PSI5_1_CH_0_GISR[DPR_RDY] | PSI5_1_CH_0_GISR[IS_OWSM[6:1]] | PSI5_1_CH_0_GISR[IS_NVSM[6:1]]
626		PSI5_1_CH0_2	PSI5_1_CH_0_NDSR[NDS31] | PSI5_1_CH_0_NDSR[NDS30] | … | PSI5_1_CH_0_NDSR[NDS1] | PSI5_1_CH_0_NDSR[NDS0]
627		PSI5_1_CH0_3	PSI5_1_CH_0_OWSR[NDS31] | PSI5_1_CH_0_OWSR[NDS30] | … | PSI5_1_CH_0_OWSR[NDS1] | PSI5_1_CH_0_OWSR[NDS0]
628		PSI5_1_CH0_4	PSI5_1_CH_0_EISR[NDS31] | PSI5_1_CH_0_EISR[NDS30] | … | PSI5_1_CH_0_EISR[NDS1] | PSI5_1_CH_0_EISR[NDS0]
629		PSI5_1_CH0_5	PSI5_1_CH_0_DSR[*] | PSI5_1_CH_0_GISR[*] | PSI5_1_CH_0_NDSR[31:0] | PSI5_1_CH_0_OWSR[31:0] | PSI5_1_CH_0_EISR[31:0]
630		PSI5_0_CH2_0	PSI5_0_CH_2_DSR[IS_DMA_TF_PM_DS] | PSI5_0_CH_2_DSR[IS_DMA_TF_SF] | PSI5_0_CH_2_DSR[IS_DMA_PM_DS_FIFO_FULL] | PSI5_0_CH_2_DSR[IS_DMA_SFUF] | PSI5_0_CH_2_DSR[IS_DMA_PM_DS_UF]
631		PSI5_0_CH2_1	PSI5_0_CH_2_GISR[IS_CESM[6:1]] | PSI5_0_CH_2_GISR[IS_STS] | PSI5_0_CH_2_GISR[IS_DTS] | PSI5_0_CH_2_GISR[[IS_DSROW] | PSI5_0_CH_2_GISR[IS_BROW] | PSI5_0_CH_2_GISR[[DSR_RDY] | PSI5_0_CH_2_GISR[DBR_RDY] | PSI5_0_CH_2_GISR[DPR_RDY] | PSI5_0_CH_2_GISR[IS_OWSM[6:1]] | PSI5_0_CH_2_GISR[IS_NVSM[6:1]]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
632		PSI5_0_CH2_2	PSI5_0_CH_2_NDSR[NDS31] | PSI5_0_CH_2_NDSR[NDS30] | … | PSI5_0_CH_2_NDSR[NDS1] | PSI5_0_CH_2_NDSR[NDS0]
633		PSI5_0_CH2_3	PSI5_0_CH_2_OWSR[NDS31] | PSI5_0_CH_2_OWSR[NDS30] | … | PSI5_0_CH_2_OWSR[NDS1] | PSI5_0_CH_2_OWSR[NDS0]
634		PSI5_0_CH2_4	PSI5_0_CH_2_EISR[NDS31] | PSI5_0_CH_2_EISR[NDS30] | … | PSI5_0_CH_2_EISR[NDS1] | PSI5_0_CH_2_EISR[NDS0]
635		PSI5_0_CH2_5	PSI5_0_CH_2_DSR[*] | PSI5_0_CH_2_GISR[*] | PSI5_0_CH_2_NDSR[31:0] | PSI5_0_CH_2_OWSR[31:0] | PSI5_0_CH_2_EISR[31:0]
636		PSI5_1_CH1_0	PSI5_1_CH_1_DSR[IS_DMA_TF_PM_DS] | PSI5_1_CH_1_DSR[IS_DMA_TF_SF] | PSI5_1_CH_1_DSR[IS_DMA_PM_DS_FIFO_FULL] | PSI5_1_CH_1_DSR[IS_DMA_SFUF] | PSI5_1_CH_1_DSR[IS_DMA_PM_DS_UF]
637		PSI5_1_CH1_1	PSI5_1_CH_1_GISR[IS_CESM[6:1]] | PSI5_1_CH_1_GISR[IS_STS] | PSI5_1_CH_1_GISR[IS_DTS] | PSI5_1_CH_1_GISR[[IS_DSROW] | PSI5_1_CH_1_GISR[IS_BROW] | PSI5_1_CH_1_GISR[[DSR_RDY] | PSI5_1_CH_1_GISR[DBR_RDY] | PSI5_1_CH_1_GISR[DPR_RDY] | PSI5_1_CH_1_GISR[IS_OWSM[6:1]] | PSI5_1_CH_1_GISR[IS_NVSM[6:1]]
638		PSI5_1_CH1_2	PSI5_1_CH_1_NDSR[NDS31] | PSI5_1_CH_1_NDSR[NDS30] | … | PSI5_1_CH_1_NDSR[NDS1] | PSI5_1_CH_1_NDSR[NDS0]
639		PSI5_1_CH1_3	PSI5_1_CH_1_OWSR[NDS31] | PSI5_1_CH_1_OWSR[NDS30] | … | PSI5_1_CH_1_OWSR[NDS1] | PSI5_1_CH_1_OWSR[NDS0]
640		PSI5_1_CH1_4	PSI5_1_CH_1_EISR[NDS31] | PSI5_1_CH_1_EISR[NDS30] | … | PSI5_1_CH_1_EISR[NDS1] | PSI5_1_CH_1_EISR[NDS0]
641		PSI5_1_CH1_5	PSI5_1_CH_1_DSR[*] | PSI5_1_CH_1_GISR[*] | PSI5_1_CH_1_NDSR[31:0] | PSI5_1_CH_1_OWSR[31:0] | PSI5_1_CH_1_EISR[31:0]
654		SIPI_0	SIPI_ERR[TOEn] | SIPI_ERR[TIDEn] | SIPI_ERR[ACKEn]
655		SIPI_1	SIPI_SR[GCRCE)

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
656		SIPI_2	SIPI_CSR0[RAR] | SIPI_CSR0[ACKR]
657		SIPI_3	SIPI_CSR1[RAR] | SIPI_CSR1[ACKR]
658		SIPI_4	SIPI_CSR2[RAR] | SIPI_CSR2[ACKR]
659		SIPI_5	SIPI_CSR3[RAR] | SIPI_CSR3[ACKR]
660		SIPI_6	SIPI_SR[TE0] | SIPI_SR[TE1] |SIPI_SR[TE2] | SIPI_SR[TE3] |SIPI_SR[MCR]
661		LFAST0_0	LFAST_0_TISR[TXPNGF] | LFAST_0_TISR[TXUNSF] | LFAST_0_TISR[TXICLCF] | LFAST_0_TISR[TXDTF]
662		LFAST0_1	LFAST_0_TISR[TXIEF] | LFAST_0_TISR[TXOVF]
663		LFAST0_2	LFAST_0_RISR[RXCTSF] | LFAST_0_RISR[RXDF] | LFAST_0_RISR[RXUNSF]
664		LFAST0_3	LFAST_0_RISR[RXUOF] | LFAST_0_RISR[RXMNF] | LFAST_0_RISR[RXMXF] | LFAST_0_RISR[RXUFF] | LFAST_0_RISR[RXOFF] | LFAST_0_RISR[RXSZF] | LFAST_0_RISR[RXICF] | LFAST_0_RISR[RXLCEF]
665		LFAST0_4	LFAST_0_RIISR[ICPFF] | LFAST_0_RIISR[ICPSF] | LFAST_0_RIISR[ICPRF] | LFAST_0_RIISR[ICTOF] | LFAST_0_RIISR[ICLPF] | LFAST_0_RIISR[ICCTF] | LFAST_0_RIISR[ICTDF] | LFAST_0_RIISR[ICTEF] | LFAST_0_RIISR[ICRFF] | LFAST_0_RIISR[ICRSF] | LFAST_0_RIISR[ICTFF] | LFAST_0_RIISR[ICTSF] | LFAST_0_RIISR[ICPOFF] | LFAST_0_RIISR[ICPONF]
674		JTAGM	JTAGM_SR[SPU_INT] | JTAGM_SR[Idle]
675		JDC	JDC_MSR[JIN_INT] | JDC_MSR[JOUT_INT]
677		M_TTCAN0_0	M_TTCAN_0 interrupt line0
678		M_TTCAN0_1	M_TTCAN_0 interrupt line1
679		M_TTCAN0_2	M_TTCAN_0 Register Time Mark interrupt
688		M_CAN1_0	m_can1_int0
689		M_CAN1_1	m_can1_int1
690		M_CAN2_0	m_can2_int0
691		M_CAN2_1	m_can2_int1
692		M_CAN3_0	m_can3_int0

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
693		M_CAN3_1	m_can3_int1
694		M_CAN4_0	m_can4_int0
695		M_CAN4_1	m_can4_int1
706		GTM_AEI_IRQ AEI	gtm_icm.gtm_aei_irq aei
707		GTM_ARU_IRQ[0]	gtm_icm.gtm_aru_irq[0]
708		GTM_ARU_IRQ[1]	gtm_icm.gtm_aru_irq[1]
709		GTM_ARU_IRQ[2]	gtm_icm.gtm_aru_irq[2]
710		GTM_BRC_IRQ	gtm_icm.gtm_brc_irq
711		GTM_CMP_IRQ	gtm_icm.gtm_cmp_irq
712		GTM_SPE0_IRQ	gtm_icm.gtm_spe0_irq
713		GTM_SPE1_IRQ	gtm_icm.gtm_spe1_irq
714		GTM_PSM0_IRQ[0]	gtm_icm.gtm_psm0_irq[0]
715		GTM_PSM0_IRQ[1]	gtm_icm.gtm_psm0_irq[1]
716		GTM_PSM0_IRQ[2]	gtm_icm.gtm_psm0_irq[2]
717		GTM_PSM0_IRQ[3]	gtm_icm.gtm_psm0_irq[3]
718		GTM_PSM0_IRQ[4]	gtm_icm.gtm_psm0_irq[4]
719		GTM_PSM0_IRQ[5]	gtm_icm.gtm_psm0_irq[5]
720		GTM_PSM0_IRQ[6]	gtm_icm.gtm_psm0_irq[6]
721		GTM_PSM0_IRQ[7]	gtm_icm.gtm_psm0_irq[7]
722		GTM_DPLL_IRQ[0]	gtm_icm.gtm_dpll_irq[0]
723		GTM_DPLL_IRQ[1]	gtm_icm.gtm_dpll_irq[1]
724		GTM_DPLL_IRQ[2]	gtm_icm.gtm_dpll_irq[2]
725		GTM_DPLL_IRQ[3]	gtm_icm.gtm_dpll_irq[3]
726		GTM_DPLL_IRQ[4]	gtm_icm.gtm_dpll_irq[4]
727		GTM_DPLL_IRQ[5]	gtm_icm.gtm_dpll_irq[5]
728		GTM_DPLL_IRQ[6]	gtm_icm.gtm_dpll_irq[6]
729		GTM_DPLL_IRQ[7]	gtm_icm.gtm_dpll_irq[7]
730		GTM_DPLL_IRQ[8]	gtm_icm.gtm_dpll_irq[8]
731		GTM_DPLL_IRQ[9]	gtm_icm.gtm_dpll_irq[9]
732		GTM_DPLL_IRQ[10]	gtm_icm.gtm_dpll_irq[10]
733		GTM_DPLL_IRQ[11]	gtm_icm.gtm_dpll_irq[11]
734		GTM_DPLL_IRQ[12]	gtm_icm.gtm_dpll_irq[12]
735		GTM_DPLL_IRQ[13]	gtm_icm.gtm_dpll_irq[13]
736		GTM_DPLL_IRQ[14]	gtm_icm.gtm_dpll_irq[14]
737		GTM_DPLL_IRQ[15]	gtm_icm.gtm_dpll_irq[15]
738		GTM_DPLL_IRQ[16]	gtm_icm.gtm_dpll_irq[16]
739		GTM_DPLL_IRQ[17]	gtm_icm.gtm_dpll_irq[17]
740		GTM_DPLL_IRQ[18]	gtm_icm.gtm_dpll_irq[18]
741		GTM_DPLL_IRQ[19]	gtm_icm.gtm_dpll_irq[19]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
742		GTM_DPLL_IRQ[20]	gtm_icm.gtm_dpll_irq[20]
743		GTM_DPLL_IRQ[21]	gtm_icm.gtm_dpll_irq[21]
744		GTM_DPLL_IRQ[22]	gtm_icm.gtm_dpll_irq[22]
745		GTM_DPLL_IRQ[23]	gtm_icm.gtm_dpll_irq[23]
746		GTM_DPLL_IRQ[24]	gtm_icm.gtm_dpll_irq[24]
747		GTM_DPLL_IRQ[25]	gtm_icm.gtm_dpll_irq[25]
748		GTM_DPLL_IRQ[26]	gtm_icm.gtm_dpll_irq[26]
749		GTM_TIM0_IRQ[0]	gtm_icm.gtm_tim0_irq[0]
750		GTM_TIM0_IRQ[1]	gtm_icm.gtm_tim0_irq[1]
751		GTM_TIM0_IRQ[2]	gtm_icm.gtm_tim0_irq[2]
752		GTM_TIM0_IRQ[3]	gtm_icm.gtm_tim0_irq[3]
753		GTM_TIM0_IRQ[4]	gtm_icm.gtm_tim0_irq[4]
754		GTM_TIM0_IRQ[5]	gtm_icm.gtm_tim0_irq[5]
755		GTM_TIM0_IRQ[6]	gtm_icm.gtm_tim0_irq[6]
756		GTM_TIM0_IRQ[7]	gtm_icm.gtm_tim0_irq[7]
757		GTM_TIM1_IRQ[0]	gtm_icm.gtm_tim1_irq[0]
758		GTM_TIM1_IRQ[1]	gtm_icm.gtm_tim1_irq[1]
759		GTM_TIM1_IRQ[2]	gtm_icm.gtm_tim1_irq[2]
760		GTM_TIM1_IRQ[3]	gtm_icm.gtm_tim1_irq[3]
761		GTM_TIM1_IRQ[4]	gtm_icm.gtm_tim1_irq[4]
762		GTM_TIM1_IRQ[5]	gtm_icm.gtm_tim1_irq[5]
763		GTM_TIM1_IRQ[6]	gtm_icm.gtm_tim1_irq[6]
764		GTM_TIM1_IRQ[7]	gtm_icm.gtm_tim1_irq[7]
765		GTM_TIM2_IRQ[0]	gtm_icm.gtm_tim2_irq[0]
766		GTM_TIM2_IRQ[1]	gtm_icm.gtm_tim2_irq[1]
767		GTM_TIM2_IRQ[2]	gtm_icm.gtm_tim2_irq[2]
768		GTM_TIM2_IRQ[3]	gtm_icm.gtm_tim2_irq[3]
769		GTM_TIM2_IRQ[4]	gtm_icm.gtm_tim2_irq[4]
770		GTM_TIM2_IRQ[5]	gtm_icm.gtm_tim2_irq[5]
771		GTM_TIM2_IRQ[6]	gtm_icm.gtm_tim2_irq[6]
772		GTM_TIM2_IRQ[7]	gtm_icm.gtm_tim2_irq[7]
773		GTM_TIM3_IRQ[0]	gtm_icm.gtm_tim3_irq[0]
774		GTM_TIM3_IRQ[1]	gtm_icm.gtm_tim3_irq[1]
775		GTM_TIM3_IRQ[2]	gtm_icm.gtm_tim3_irq[2]
776		GTM_TIM3_IRQ[3]	gtm_icm.gtm_tim3_irq[3]
777		GTM_TIM3_IRQ[4]	gtm_icm.gtm_tim3_irq[4]
778		GTM_TIM3_IRQ[5]	gtm_icm.gtm_tim3_irq[5]
779		GTM_TIM3_IRQ[6]	gtm_icm.gtm_tim3_irq[6]
780		GTM_TIM3_IRQ[7]	gtm_icm.gtm_tim3_irq[7]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
781		GTM_MCS0_IRQ[0]	gtm_icm.gtm_mcs0_irq[0]
782		GTM_MCS0_IRQ[1]	gtm_icm.gtm_mcs0_irq[1]
783		GTM_MCS0_IRQ[2]	gtm_icm.gtm_mcs0_irq[2]
784		GTM_MCS0_IRQ[3]	gtm_icm.gtm_mcs0_irq[3]
785		GTM_MCS0_IRQ[4]	gtm_icm.gtm_mcs0_irq[4]
786		GTM_MCS0_IRQ[5]	gtm_icm.gtm_mcs0_irq[5]
787		GTM_MCS0_IRQ[6]	gtm_icm.gtm_mcs0_irq[6]
788		GTM_MCS0_IRQ[7]	gtm_icm.gtm_mcs0_irq[7]
789		GTM_MCS1_IRQ[0]	gtm_icm.gtm_mcs1_irq[0]
790		GTM_MCS1_IRQ[1]	gtm_icm.gtm_mcs1_irq[1]
791		GTM_MCS1_IRQ[2]	gtm_icm.gtm_mcs1_irq[2]
792		GTM_MCS1_IRQ[3]	gtm_icm.gtm_mcs1_irq[3]
793		GTM_MCS1_IRQ[4]	gtm_icm.gtm_mcs1_irq[4]
794		GTM_MCS1_IRQ[5]	gtm_icm.gtm_mcs1_irq[5]
795		GTM_MCS1_IRQ[6]	gtm_icm.gtm_mcs1_irq[6]
796		GTM_MCS1_IRQ[7]	gtm_icm.gtm_mcs1_irq[7]
797		GTM_MCS2_IRQ[0]	gtm_icm.gtm_mcs2_irq[0]
798		GTM_MCS2_IRQ[1]	gtm_icm.gtm_mcs2_irq[1]
799		GTM_MCS2_IRQ[2]	gtm_icm.gtm_mcs2_irq[2]
800		GTM_MCS2_IRQ[3]	gtm_icm.gtm_mcs2_irq[3]
801		GTM_MCS2_IRQ[4]	gtm_icm.gtm_mcs2_irq[4]
802		GTM_MCS2_IRQ[5]	gtm_icm.gtm_mcs2_irq[5]
803		GTM_MCS2_IRQ[6]	gtm_icm.gtm_mcs2_irq[6]
804		GTM_MCS2_IRQ[7]	gtm_icm.gtm_mcs2_irq[7]
805		GTM_MCS3_IRQ[0]	gtm_icm.gtm_mcs3_irq[0]
806		GTM_MCS3_IRQ[1]	gtm_icm.gtm_mcs3_irq[1]
807		GTM_MCS3_IRQ[2]	gtm_icm.gtm_mcs3_irq[2]
808		GTM_MCS3_IRQ[3]	gtm_icm.gtm_mcs3_irq[3]
809		GTM_MCS3_IRQ[4]	gtm_icm.gtm_mcs3_irq[4]
810		GTM_MCS3_IRQ[5]	gtm_icm.gtm_mcs3_irq[5]
811		GTM_MCS3_IRQ[6]	gtm_icm.gtm_mcs3_irq[6]
812		GTM_MCS3_IRQ[7]	gtm_icm.gtm_mcs3_irq[7]
813		GTM_TOM0_IRQ[0]	gtm_icm.gtm_tom0_irq[0]
814		GTM_TOM0_IRQ[1]	gtm_icm.gtm_tom0_irq[1]
815		GTM_TOM0_IRQ[2]	gtm_icm.gtm_tom0_irq[2]
816		GTM_TOM0_IRQ[3]	gtm_icm.gtm_tom0_irq[3]
817		GTM_TOM0_IRQ[4]	gtm_icm.gtm_tom0_irq[4]
818		GTM_TOM0_IRQ[5]	gtm_icm.gtm_tom0_irq[5]
819		GTM_TOM0_IRQ[6]	gtm_icm.gtm_tom0_irq[6]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
820		GTM_TOM0_IRQ[7]	gtm_icm.gtm_tom0_irq[7]
821		GTM_TOM1_IRQ[0]	gtm_icm.gtm_tom1_irq[0]
822		GTM_TOM1_IRQ[1]	gtm_icm.gtm_tom1_irq[1]
823		GTM_TOM1_IRQ[2]	gtm_icm.gtm_tom1_irq[2]
824		GTM_TOM1_IRQ[3]	gtm_icm.gtm_tom1_irq[3]
825		GTM_TOM1_IRQ[4]	gtm_icm.gtm_tom1_irq[4]
826		GTM_TOM1_IRQ[5]	gtm_icm.gtm_tom1_irq[5]
827		GTM_TOM1_IRQ[6]	gtm_icm.gtm_tom1_irq[6]
828		GTM_TOM1_IRQ[7]	gtm_icm.gtm_tom1_irq[7]
829		GTM_TOM2_IRQ[0]	gtm_icm.gtm_tom2_irq[0]
830		GTM_TOM2_IRQ[1]	gtm_icm.gtm_tom2_irq[1]
831		GTM_TOM2_IRQ[2]	gtm_icm.gtm_tom2_irq[2]
832		GTM_TOM2_IRQ[3]	gtm_icm.gtm_tom2_irq[3]
833		GTM_TOM2_IRQ[4]	gtm_icm.gtm_tom2_irq[4]
834		GTM_TOM2_IRQ[5]	gtm_icm.gtm_tom2_irq[5]
835		GTM_TOM2_IRQ[6]	gtm_icm.gtm_tom2_irq[6]
836		GTM_TOM2_IRQ[7]	gtm_icm.gtm_tom2_irq[7]
837		GTM_ATOM0_IRQ[0]	gtm_icm.gtm_atom0_irq[0]
838		GTM_ATOM0_IRQ[1]	gtm_icm.gtm_atom0_irq[1]
839		GTM_ATOM0_IRQ[2]	gtm_icm.gtm_atom0_irq[2]
840		GTM_ATOM0_IRQ[3]	gtm_icm.gtm_atom0_irq[3]
841		GTM_ATOM1_IRQ[0]	gtm_icm.gtm_atom1_irq[0]
842		GTM_ATOM1_IRQ[1]	gtm_icm.gtm_atom1_irq[1]
843		GTM_ATOM1_IRQ[2]	gtm_icm.gtm_atom1_irq[2]
844		GTM_ATOM1_IRQ[3]	gtm_icm.gtm_atom1_irq[3]
845		GTM_ATOM2_IRQ[0]	gtm_icm.gtm_atom2_irq[0]
846		GTM_ATOM2_IRQ[1]	gtm_icm.gtm_atom2_irq[1]
847		GTM_ATOM2_IRQ[2]	gtm_icm.gtm_atom2_irq[2]
848		GTM_ATOM2_IRQ[3]	gtm_icm.gtm_atom2_irq[3]
849		GTM_ATOM3_IRQ[0]	gtm_icm.gtm_atom3_irq[0]
850		GTM_ATOM3_IRQ[1]	gtm_icm.gtm_atom3_irq[1]
851		GTM_ATOM3_IRQ[2]	gtm_icm.gtm_atom3_irq[2]
852		GTM_ATOM3_IRQ[3]	gtm_icm.gtm_atom3_irq[3]
853		GTM_ATOM4_IRQ[0]	gtm_icm.gtm_atom4_irq[0]
854		GTM_ATOM4_IRQ[1]	gtm_icm.gtm_atom4_irq[1]
855		GTM_ATOM4_IRQ[2]	gtm_icm.gtm_atom4_irq[2]
856		GTM_ATOM4_IRQ[3]	gtm_icm.gtm_atom4_irq[3]
857		GTM_SPE2_IRQ	gtm_icm.gtm_spe2_irq
858		GTM_SPE3_IRQ	gtm_icm.gtm_spe3_irq

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
859		GTM_PSM1_IRQ[0]	gtm_icm.gtm_psm1_irq[0]
860		GTM_PSM1_IRQ[1]	gtm_icm.gtm_psm1_irq[1]
861		GTM_PSM1_IRQ[2]	gtm_icm.gtm_psm1_irq[2]
862		GTM_PSM1_IRQ[3]	gtm_icm.gtm_psm1_irq[3]
863		GTM_PSM1_IRQ[4]	gtm_icm.gtm_psm1_irq[4]
864		GTM_PSM1_IRQ[5]	gtm_icm.gtm_psm1_irq[5]
865		GTM_PSM1_IRQ[6]	gtm_icm.gtm_psm1_irq[6]
866		GTM_PSM1_IRQ[7]	gtm_icm.gtm_psm1_irq[7]
867		GTM_TIM4_IRQ[0]	gtm_icm.gtm_tim4_irq[0]
868		GTM_TIM4_IRQ[1]	gtm_icm.gtm_tim4_irq[1]
869		GTM_TIM4_IRQ[2]	gtm_icm.gtm_tim4_irq[2]
870		GTM_TIM4_IRQ[3]	gtm_icm.gtm_tim4_irq[3]
871		GTM_TIM4_IRQ[4]	gtm_icm.gtm_tim4_irq[4]
872		GTM_TIM4_IRQ[5]	gtm_icm.gtm_tim4_irq[5]
873		GTM_TIM4_IRQ[6]	gtm_icm.gtm_tim4_irq[6]
874		GTM_TIM4_IRQ[7]	gtm_icm.gtm_tim4_irq[7]
875		GTM_TIM5_IRQ[0]	gtm_icm.gtm_tim5_irq[0]
876		GTM_TIM5_IRQ[1]	gtm_icm.gtm_tim5_irq[1]
877		GTM_TIM5_IRQ[2]	gtm_icm.gtm_tim5_irq[2]
878		GTM_TIM5_IRQ[3]	gtm_icm.gtm_tim5_irq[3]
879		GTM_TIM5_IRQ[4]	gtm_icm.gtm_tim5_irq[4]
880		GTM_TIM5_IRQ[5]	gtm_icm.gtm_tim5_irq[5]
881		GTM_TIM5_IRQ[6]	gtm_icm.gtm_tim5_irq[6]
882		GTM_TIM5_IRQ[7]	gtm_icm.gtm_tim5_irq[7]
883		GTM_MCS4_IRQ[0]	gtm_icm.gtm_mcs4_irq[0]
884		GTM_MCS4_IRQ[1]	gtm_icm.gtm_mcs4_irq[1]
885		GTM_MCS4_IRQ[2]	gtm_icm.gtm_mcs4_irq[2]
886		GTM_MCS4_IRQ[3]	gtm_icm.gtm_mcs4_irq[3]
887		GTM_MCS4_IRQ[4]	gtm_icm.gtm_mcs4_irq[4]
888		GTM_MCS4_IRQ[5]	gtm_icm.gtm_mcs4_irq[5]
889		GTM_MCS4_IRQ[6]	gtm_icm.gtm_mcs4_irq[6]
890		GTM_MCS4_IRQ[7]	gtm_icm.gtm_mcs4_irq[7]
891		GTM_MCS5_IRQ[0]	gtm_icm.gtm_mcs5_irq[0]
892		GTM_MCS5_IRQ[1]	gtm_icm.gtm_mcs5_irq[1]
893		GTM_MCS5_IRQ[2]	gtm_icm.gtm_mcs5_irq[2]
894		GTM_MCS5_IRQ[3]	gtm_icm.gtm_mcs5_irq[3]
895		GTM_MCS5_IRQ[4]	gtm_icm.gtm_mcs5_irq[4]
896		GTM_MCS5_IRQ[5]	gtm_icm.gtm_mcs5_irq[5]
897		GTM_MCS5_IRQ[6]	gtm_icm.gtm_mcs5_irq[6]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
898		GTM_MCS5_IRQ[7]	gtm_icm.gtm_mcs5_irq[7]
899		GTM_TOM3_IRQ[0]	gtm_icm.gtm_tom3_irq[0]
900		GTM_TOM3_IRQ[1]	gtm_icm.gtm_tom3_irq[1]
901		GTM_TOM3_IRQ[2]	gtm_icm.gtm_tom3_irq[2]
902		GTM_TOM3_IRQ[3]	gtm_icm.gtm_tom3_irq[3]
903		GTM_TOM3_IRQ[4]	gtm_icm.gtm_tom3_irq[4]
904		GTM_TOM3_IRQ[5]	gtm_icm.gtm_tom3_irq[5]
905		GTM_TOM3_IRQ[6]	gtm_icm.gtm_tom3_irq[6]
906		GTM_TOM3_IRQ[7]	gtm_icm.gtm_tom3_irq[7]
907		GTM_TOM4_IRQ[0]	gtm_icm.gtm_tom4_irq[0]
908		GTM_TOM4_IRQ[1]	gtm_icm.gtm_tom4_irq[1]
909		GTM_TOM4_IRQ[2]	gtm_icm.gtm_tom4_irq[2]
910		GTM_TOM4_IRQ[3]	gtm_icm.gtm_tom4_irq[3]
911		GTM_TOM4_IRQ[4]	gtm_icm.gtm_tom4_irq[4]
912		GTM_TOM4_IRQ[5]	gtm_icm.gtm_tom4_irq[5]
913		GTM_TOM4_IRQ[6]	gtm_icm.gtm_tom4_irq[6]
914		GTM_TOM4_IRQ[7]	gtm_icm.gtm_tom4_irq[7]
915		GTM_ATOM5_IRQ[0]	gtm_icm.gtm_atom5_irq[0]
916		GTM_ATOM5_IRQ[1]	gtm_icm.gtm_atom5_irq[1]
917		GTM_ATOM5_IRQ[2]	gtm_icm.gtm_atom5_irq[2]
918		GTM_ATOM5_IRQ[3]	gtm_icm.gtm_atom5_irq[3]
919		GTM_ATOM6_IRQ[0]	gtm_icm.gtm_atom6_irq[0]
920		GTM_ATOM6_IRQ[1]	gtm_icm.gtm_atom6_irq[1]
921		GTM_ATOM6_IRQ[2]	gtm_icm.gtm_atom6_irq[2]
922		GTM_ATOM6_IRQ[3]	gtm_icm.gtm_atom6_irq[3]
923		GTM_ATOM7_IRQ[0]	gtm_icm.gtm_atom7_irq[0]
924		GTM_ATOM7_IRQ[1]	gtm_icm.gtm_atom7_irq[1]
925		GTM_ATOM7_IRQ[2]	gtm_icm.gtm_atom7_irq[2]
926		GTM_ATOM7_IRQ[3]	gtm_icm.gtm_atom7_irq[3]
927		GTM_ATOM8_IRQ[0]	gtm_icm.gtm_atom8_irq[0]
928		GTM_ATOM8_IRQ[1]	gtm_icm.gtm_atom8_irq[1]
929		GTM_ATOM8_IRQ[2]	gtm_icm.gtm_atom8_irq[2]
930		GTM_ATOM8_IRQ[3]	gtm_icm.gtm_atom8_irq[3]
931		GTM_ERR_IRQ	gtm_err_irq
946		PS_SR_IRQ[0]	PS_MBOX_SR_IRQ[0] PS_ERR_SR_IRQ[0]
947		PS_SR_IRQ[1]	PS_MBOX_SR_IRQ[1] PS_ERR_SR_IRQ[1]
948		PS_SR_IRQ[2]	PS_MBOX_SR_IRQ[2]

Table continues on the next page...

Table 7-23. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
			PS_ERR_SR_IRQ[2]
949		PS_SR_IRQ[3]	PS_MBOX_SR_IRQ[3] PS_ERR_SR_IRQ[3]
950		PS_SR_IRQ[4]	PS_MBOX_SR_IRQ[4] PS_ERR_SR_IRQ[4]
951		PS_SR_IRQ[5]	PS_MBOX_SR_IRQ[5] PS_ERR_SR_IRQ[5]
952		PS_SR_IRQ[6]	PS_MBOX_SR_IRQ[6] PS_ERR_SR_IRQ[6]
953		PS_SR_IRQ[7]	PS_MBOX_SR_IRQ[7] PS_ERR_SR_IRQ[7]
954		PSI5_E2SSR[1]	PSI5_E2SSR[1]
955		PSI5_E2SSR[2]	PSI5_E2SSR[2]
956		PSI5_E2SSR[3]	PSI5_E2SSR[3]
957		PSI5_E2SSR[4]	PSI5_E2SSR[4]
958		PSI5_E2SSR[5]	PSI5_E2SSR[5]
959		PSI5_E2SSR[6]	PSI5_E2SSR[6]
960		PSI5_E2SSR[7]	PSI5_E2SSR[7]
961		PS_GLSR	PS_GLSR
962		PSI5_S_0_UART_RX	PSI5_S_0_UART_RX
963		PSI5_S_0_UART_TX	PSI5_S_0_UART_TX
964		PSI5_S_0_UART_ERR	PSI5_S_0_UART_ERR

7.3.7.4 Interrupt Latency

The Interrupt systems is designed to provide a very short latency between the assertion of an interrupt request from a peripheral and the first instruction of the Interrupt Service Routine.

The latency time is difficult to define exactly because it depends what interrupts the INTC is processing and what the CPU is doing.

But in general, the latency time is divided into 3 sub-sections:

- INTC Latency
  - 3 clock cycles (INTC clock domain) from the assertion of the interrupt request from the peripheral to assertion of the interrupt request to the CPU
- CPU Latency

- 4 clock cycles (CPU clock domain) from the assertion of the interrupt request from the INTC to the execution of the first instruction

- Software Latency
  - 7 clock cycles (CPU clock domain) to execute the 3 instructions needed to save essential registers and enable interrupts

For the MPC5777M, the INTC operates at 50 MHz and the CPUs operate at 300 MHz. So interrupt latency is:-

INTC Latency = 60 ns

CPU Latency = 13.33 ns

Software Latency = 23.33 ns

Interrupt Latency Total = 96.66 ns

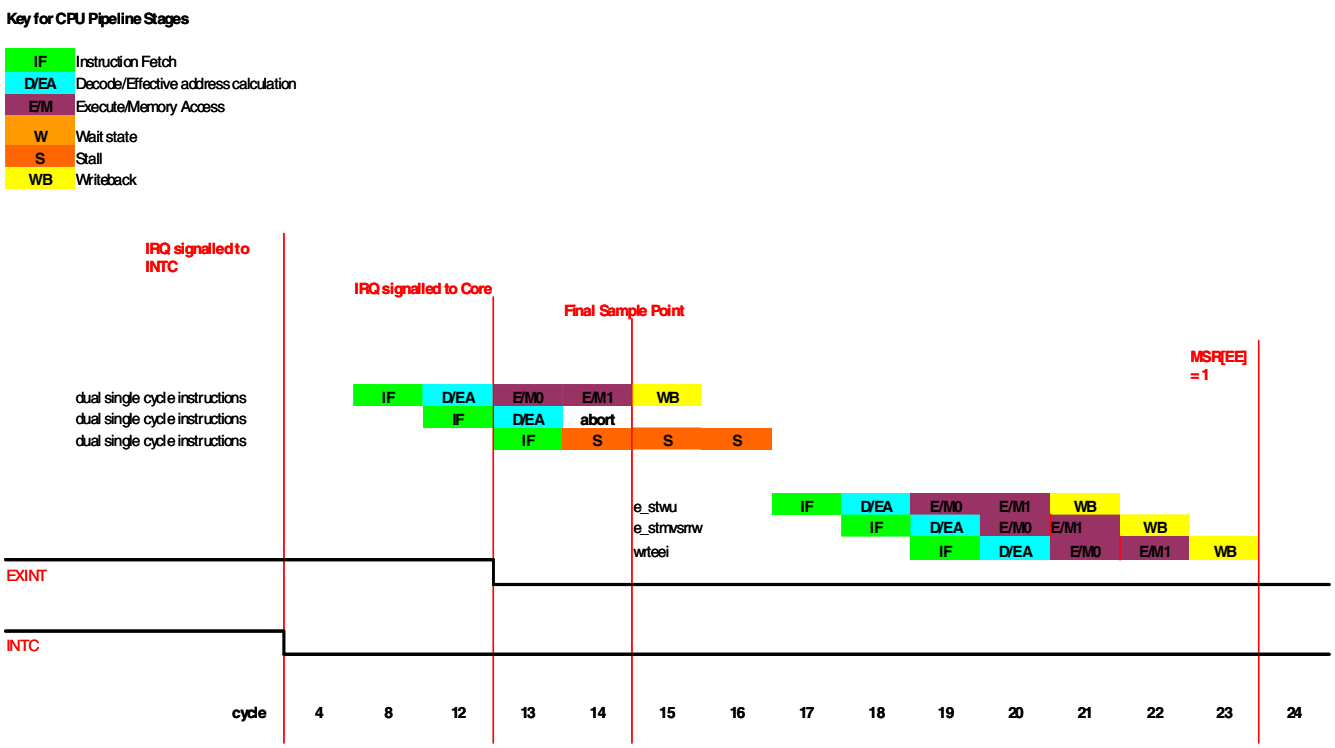


Figure 7-13. Interrupt timing

7.3.7.5 INTC (hsm_z0_platform) configuration

For interrupts specific to the Hardware Security Module (HSM) please refer to the *MPC5777M Security Reference Manual*.

7.4 DMA Controller configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

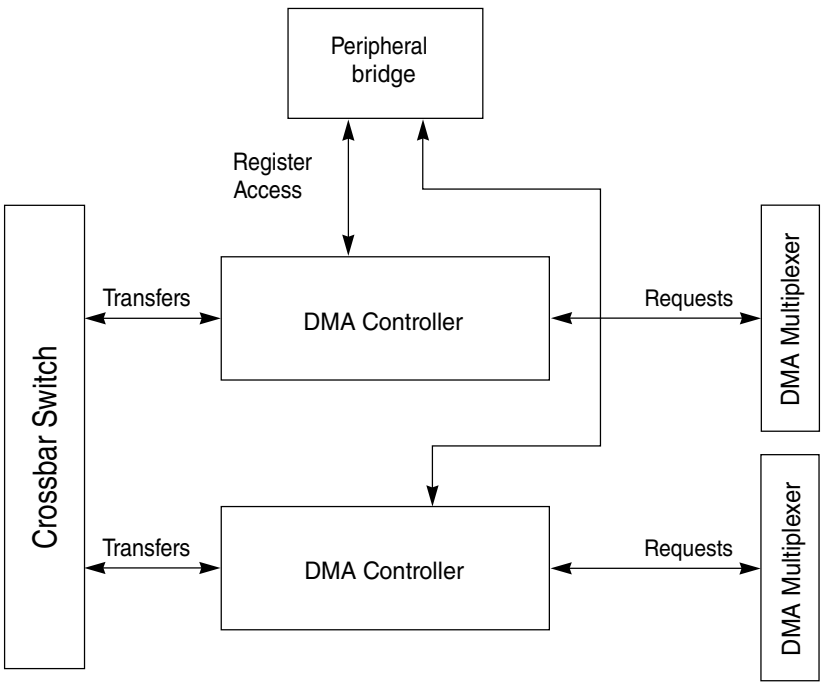


Figure 7-14. DMA controller configuration

Table 7-24. Reference links to related information

Topic	Related module	Reference
Full description	DMA Controller	Enhanced Direct Memory Access (eDMA)
System memory map		Memory Map
Register access	Peripheral bridge	Peripheral Bridge
Clocking		Clocking
Transfers	Crossbar switch	Crossbar Switch (XBAR)

7.5 DMACHMUX configuration

The peripheral DMA requests are serviced by a DMA controller. A static multiplexing scheme allows the support of more peripheral requests than available DMA channels.

7.5.1 DMA channel assignment

MPC5777M has ten multiplexers and a total of 128 DMA channels. Since more than 64 DMA channels are used, two eDMA modules (eDMA_0 and eDMA_1) are required. Table 7-25 lists which multiplexers and channels are available on MPC5777M.

Table 7-25. MPC5777M multiplexer and channel list

DMAMUX instance	Number of DMA channels	DMAMUX channel	eDMA_0 channel	eDMA_1 channel
0	8	0–7	0–7	
1	8	8–15	8–15	
2	8	16–23	16–23	
3	8	24–31	24–31	
4	16	32–47	32–47	
5	16	48–63	48–63	
6	16	64–79		0-15
7	16	80–95		16-31
8	16	96–111		32-47
9	16	112–127		48-63

7.5.2 DMA system level block diagram

The connections of the DMA channel multiplexers to the eDMA are shown in Figure 7-15.

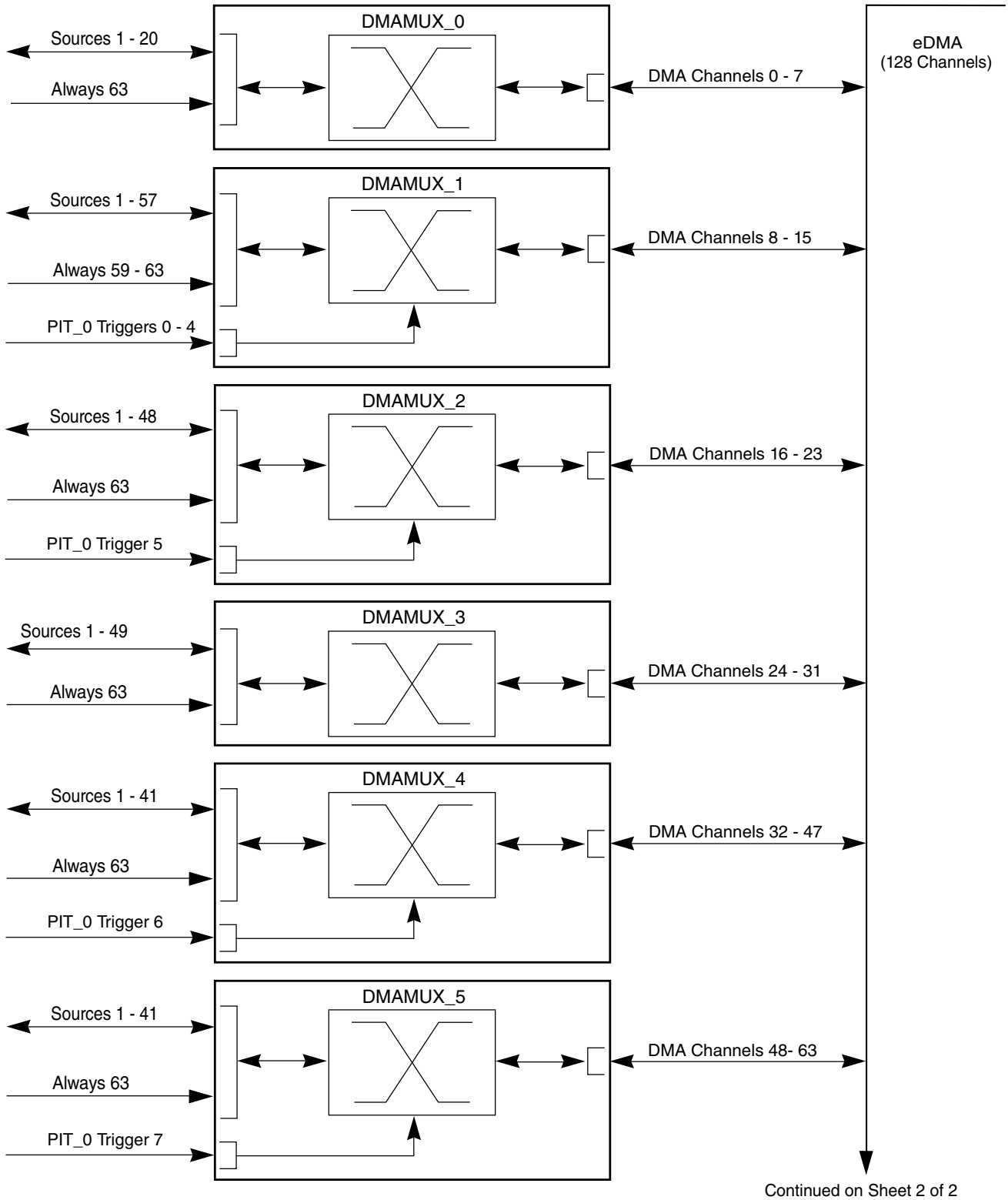


Figure 7-15. DMA channel multiplexing (Sheet 1 of 2)

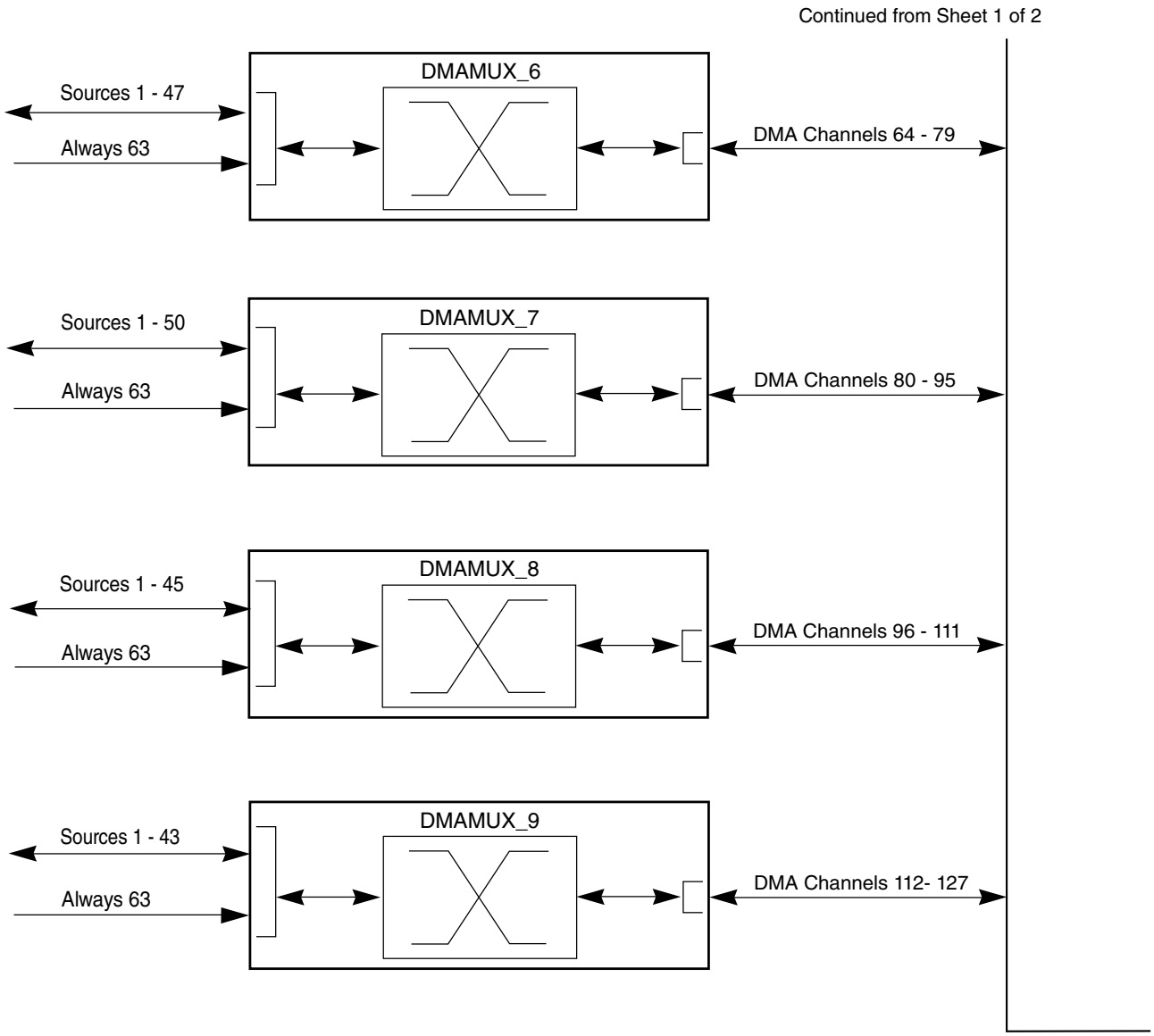


Figure 7-16. DMA channel multiplexing (Sheet 2 of 2)

7.5.3 DMA request mapping

The mapping of the peripheral DMA requests to the DMA_CH_MUX input sources is shown in Table 7-26.

The DMA requests from the Sigma-Delta Analog-to-Digital Converter (SDADC) must go through a two input AND gate before the DMAMUX. The other source of the AND gate is an IMUX output of GTM channels. The exact IMUX output is detailed in the SIUL2 chapter.

Table 7-26. DMAMUX peripheral DMA request to input source mapping

SOURCE	Peripheral DMA requests									
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3	DMAMUX_4	DMAMUX_5	DMAMUX_6	DMAMUX_7	DMAMUX_8	DMAMUX_9
0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
1	ADC_SAR_0 EOC	DSPI_12 RX	ADC_SAR_2 EOC	ADC_SAR_3 EOC	ADC_SAR_4 EOC	ADC_SAR_7 EOC	ADC_SAR_8 EOC	GTM_MCS4_IRQ0	GTM_TIM4_IRQ0	GTM_PSM1_IRQ0
2	ADC_SAR_1 EOC	DSPI_12 TX	ADC_SD_1 EOC	DSPI_2 RX	ADC_SAR_6 EOC	ADC_SD_4 EOC	ADC_SAR_9 EOC	GTM_MCS4_IRQ1	GTM_TIM4_IRQ1	GTM_PSM1_IRQ1
3	ADC_SAR_B EOC	LINFlexD_0 RX	DSPI_1 RX	DSPI_2 TX	ADC_SD_2 EOC	ADC_SD_5 EOC	ADC_SAR_10 EOC	GTM_MCS4_IRQ2	GTM_TIM4_IRQ2	GTM_PSM1_IRQ2
4	ADC_SD_0 EOC	LINFlexD_0 TX	DSPI_1 TX	LINFlexD_2 RX	ADC_SD_3 EOC	DSPI_5 RX	ADC_SAR_5 EOC	GTM_MCS4_IRQ3	GTM_TIM4_IRQ3	GTM_PSM1_IRQ3
5	DSPI_0 RX	LINFlexD_1 RX	SENT_1 RX FAST	LINFlexD_2 TX	DSPI_3 RX	DSPI_5 TX	ADC_SD_6 EOC	GTM_MCS4_IRQ4	GTM_TIM4_IRQ4	GTM_PSM1_IRQ4
6	DSPI_0 TX	LINFlexD_1 TX	SENT_1 RX SLOW	I2C_0 RX	DSPI_3 TX	LINFlexD_1 RX	ADC_SD_7 EOC	GTM_MCS4_IRQ5	GTM_TIM4_IRQ5	GTM_PSM1_IRQ5
7	DSPI_4 RX	LINFlexD_14 RX	PSI5_0 CH0 RX PSI5	I2C_0 TX	LINFlexD_0 RX	LINFlexD_1 TX	ADC_SD_8 EOC	GTM_MCS4_IRQ6	GTM_TIM4_IRQ6	GTM_PSM1_IRQ6
8	DSPI_4 Tx	LINFlexD_14 TX	PSI5_0 CH0 RX SENT	PSI5_1 CH0 RX PSI5	LINFlexD_0 TX	LINFlexD_15 RX	ADC_SD_9 EOC	GTM_MCS4_IRQ7	GTM_TIM4_IRQ7	GTM_PSM1_IRQ7
9	Reserved	SENT_0 RX FAST	SIUL2 REQ2	PSI5_1 CH0 RX SENT	LINFlexD_14 RX	LINFlexD_15 TX	PSI5_0 CH2 RX PSI5	GTM_MCS5_IRQ0	GTM_TIM5_IRQ0	GTM_SPE2
10	ADC_SAR_4 EOC	SENT_0 RX SLOW	SIUL2 REQ4	SIUL2 REQ5	LINFlexD_14 TX	SENT_0 RX FAST	PSI5_0 CH2 RX SENT	GTM_MCS5_IRQ1	GTM_TIM5_IRQ1	GTM_SPE3
11	ADC_SD_3 EOC	SIPI CH0	GTM_PSM0_IRQ0	GTM_PSM0_IRQ4	PSI5_0 CH1 RX PSI5	SENT_0 RX SLOW	PSI5_1 CH1 RX PSI5	GTM_MCS5_IRQ2	GTM_TIM5_IRQ2	I2C_0 RX
12	MCAN_1	SIPI CH1	GTM_PSM0_IRQ1	GTM_PSM0_IRQ5	PSI5_0 CH1 RX SENT	SIPI CH2	PSI5_1 CH1 RX SENT	GTM_MCS5_IRQ3	GTM_TIM5_IRQ3	I2C_0 TX
13	MCAN_2	SIPI CH2	GTM_PSM0_IRQ2	GTM_PSM0_IRQ6	SIPI CH0	SIPI CH3	DSPI_6 RX	GTM_MCS5_IRQ4	GTM_TIM5_IRQ4	GTM_ATOM5_IRQ0
14	SENT_0 RX FAST	SIPI CH3	GTM_PSM0_IRQ3	GTM_PSM0_IRQ7	SIPI CH1	SIUL2 REQ10	DSPI_6 TX	GTM_MCS5_IRQ5	GTM_TIM5_IRQ5	GTM_ATOM5_IRQ1
15	SENT_0 RX SLOW	SIUL2 REQ0	GTM_TIM1_IRQ0	GTM_TIM1_IRQ4	SIUL2 REQ9	GTM_TIM0_IRQ2	DSPI_6 CMD	GTM_MCS5_IRQ6	GTM_TIM5_IRQ6	GTM_ATOM7_IRQ0
16	LINFlexD_0 RX	SIUL2 REQ1	GTM_TIM1_IRQ1	GTM_TIM1_IRQ5	GTM_TIM0_IRQ0	GTM_TIM0_IRQ3	LINFlexD_16 RX	GTM_MCS5_IRQ7	GTM_TIM5_IRQ7	GTM_ATOM7_IRQ1

Table continues on the next page...

Table 7-26. DMAMUX peripheral DMA request to input source mapping (continued)

SOURCE	Peripheral DMA requests									
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3	DMAMUX_4	DMAMUX_5	DMAMUX_6	DMAMUX_7	DMAMUX_8	DMAMUX_9
17	LINFlexD_0 TX	GTM_TIM0_IRQ0	GTM_TIM1_IRQ2	GTM_TIM1_IRQ6	GTM_TIM0_IRQ1	GTM_TOM0_IRQ4	LINFlexD_16 TX	GTM_ATOM5_IRQ0	GTM_ATOM7_IRQ0	GTM_TOM3_IRQ0
18	LINFlexD_14 RX	GTM_TIM0_IRQ1	GTM_TIM1_IRQ3	GTM_TIM1_IRQ7	GTM_TOM0_IRQ0	GTM_TOM0_IRQ5	I2C_1 RX	GTM_ATOM5_IRQ1	GTM_ATOM7_IRQ1	GTM_TOM3_IRQ1
19	DSPI_0 CMD	GTM_TIM0_IRQ2	GTM_TOM1_IRQ0	GTM_TOM1_IRQ4	GTM_TOM0_IRQ1	GTM_PSM0_IRQ4	I2C_1 TX	GTM_ATOM5_IRQ2	GTM_ATOM7_IRQ2	GTM_TOM3_IRQ2
20	DSPI_4 CMD	GTM_TIM0_IRQ3	GTM_TOM1_IRQ1	GTM_TOM1_IRQ5	GTM_PSM0_IRQ0	GTM_PSM0_IRQ5	GTM_PSM1_IRQ0	GTM_ATOM5_IRQ3	GTM_ATOM7_IRQ3	GTM_TOM3_IRQ3
21	Reserved	GTM_TIM0_IRQ4	GTM_TOM1_IRQ2	GTM_TOM1_IRQ6	GTM_PSM0_IRQ1	GTM_PSM0_IRQ6	GTM_PSM1_IRQ1	GTM_ATOM6_IRQ0	GTM_ATOM8_IRQ0	GTM_TOM3_IRQ4
22	Reserved	GTM_TIM0_IRQ5	GTM_TOM1_IRQ3	GTM_TOM1_IRQ7	GTM_PSM0_IRQ2	GTM_PSM0_IRQ7	GTM_PSM1_IRQ2	GTM_ATOM6_IRQ1	GTM_ATOM8_IRQ1	GTM_TOM3_IRQ5
23	Reserved	GTM_TIM0_IRQ6	GTM_ATOM1_IRQ0	GTM_ATOM1_IRQ2	GTM_PSM0_IRQ3	GTM_TOM1_IRQ4	GTM_PSM1_IRQ3	GTM_ATOM6_IRQ2	GTM_ATOM8_IRQ2	GTM_TOM3_IRQ6
24	Reserved	GTM_TIM0_IRQ7	GTM_ATOM1_IRQ1	GTM_ATOM1_IRQ3	GTM_TOM1_IRQ0	GTM_TOM1_IRQ5	DSPI_1 CMD	GTM_ATOM6_IRQ3	GTM_ATOM8_IRQ3	GTM_TOM3_IRQ7
25	Reserved	GTM_TOM0_IRQ0	GTM_MCS1_IRQ0	GTM_MCS1_IRQ4	GTM_TOM1_IRQ1	GTM_TIM3_IRQ4	DSPI_1 RX	GTM_TOM3_IRQ0	GTM_TOM4_IRQ0	GTM_TOM4_IRQ0
26	Reserved	GTM_TOM0_IRQ1	GTM_MCS1_IRQ1	GTM_MCS1_IRQ5	GTM_TIM3_IRQ0	GTM_TIM3_IRQ5	DSPI_1 TX	GTM_TOM3_IRQ1	GTM_TOM4_IRQ1	GTM_TOM4_IRQ1
27	Reserved	GTM_TOM0_IRQ2	GTM_MCS1_IRQ2	GTM_MCS1_IRQ6	GTM_TIM3_IRQ1	GTM_TIM3_IRQ6	DSPI_12 CMD	GTM_TOM3_IRQ2	GTM_TOM4_IRQ2	GTM_TOM4_IRQ2
28	Reserved	GTM_TOM0_IRQ3	GTM_MCS1_IRQ3	GTM_MCS1_IRQ7	GTM_TIM3_IRQ2	GTM_TIM3_IRQ7	DSPI_12 RX	GTM_TOM3_IRQ3	GTM_TOM4_IRQ3	GTM_TOM4_IRQ3
29	Reserved	GTM_TOM0_IRQ4	GTM_TIM2_IRQ0	GTM_TIM2_IRQ4	GTM_TIM3_IRQ3	GTM_MCS3_IRQ4	DSPI_12 TX	GTM_TOM3_IRQ4	GTM_TOM4_IRQ4	GTM_TOM4_IRQ4
30	Reserved	GTM_TOM0_IRQ5	GTM_TIM2_IRQ1	GTM_TIM2_IRQ5	GTM_MCS3_IRQ0	GTM_MCS3_IRQ5	SENT_1 RX FAST	GTM_TOM3_IRQ5	GTM_TOM4_IRQ5	GTM_TOM4_IRQ5
31	Reserved	GTM_TOM0_IRQ6	GTM_TIM2_IRQ2	GTM_TIM2_IRQ6	GTM_MCS3_IRQ1	GTM_MCS3_IRQ6	SENT_1 RX SLOW	GTM_TOM3_IRQ6	GTM_TOM4_IRQ6	GTM_TOM4_IRQ6
32	Reserved	GTM_TOM0_IRQ7	GTM_TIM2_IRQ3	GTM_TIM2_IRQ7	GTM_MCS3_IRQ2	GTM_MCS3_IRQ7	GTM_TOM1_IRQ0	GTM_TOM3_IRQ7	GTM_TOM4_IRQ7	GTM_TOM4_IRQ7
33	Reserved	GTM_ATOM0_IRQ0	GTM_ATOM2_IRQ0	GTM_ATOM2_IRQ2	GTM_MCS3_IRQ3	DSPI_5 CMD	GTM_TOM1_IRQ1	ADC_SD_6 EOC	ADC_SD_8 EOC	ADC_SD_9 EOC

Table continues on the next page...

Table 7-26. DMAMUX peripheral DMA request to input source mapping (continued)

SOURCE	Peripheral DMA requests									
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3	DMAMUX_4	DMAMUX_5	DMAMUX_6	DMAMUX_7	DMAMUX_8	DMAMUX_9
34	Reserved	GTM_ATOM0_IRQ1	GTM_ATOM2_IRQ1	GTM_ATOM2_IRQ3	DSPI_3 CMD	MCAN_1	GTM_TOM1_IRQ2	ADC_SD_7 EOC	PSI5_1 CH0 RX PSI5	MCAN_1
35	Reserved	GTM_ATOM0_IRQ2	GTM_MCS2_IRQ0	GTM_MCS2_IRQ4	ADC_SD_1 EOC	MCAN_2	GTM_TOM1_IRQ3	PSI5_0 CH0 RX PSI5	PSI5_1 CH0 RX SENT	MCAN_2
36	Reserved	GTM_ATOM0_IRQ3	GTM_MCS2_IRQ1	GTM_MCS2_IRQ5	ADC_SD_4 EOC	DSPI_3 RX	GTM_TOM1_IRQ4	PSI5_0 CH0 RX SENT	PSI5_1 CH1 RX PSI5	ADC_SAR_3 EOC
37	Reserved	GTM_MCS0_IRQ0	GTM_MCS2_IRQ2	GTM_MCS2_IRQ6	ADC_SD_5 EOC	DSPI_3 TX	GTM_TOM1_IRQ5	PSI5_0 CH1 RX PSI5	PSI5_1 CH1 RX SENT	ADC_SAR_4 EOC
38	Reserved	GTM_MCS0_IRQ1	GTM_MCS2_IRQ3	GTM_MCS2_IRQ7	ADC_SAR_0 EOC	ADC_SAR_3 EOC	GTM_TOM1_IRQ6	PSI5_0 CH1 RX SENT	ADC_SAR_0 EOC	ADC_SAR_6 EOC
39	Reserved	GTM_MCS0_IRQ2	GTM_ATOM3_IRQ0	GTM_ATOM3_IRQ2	DSPI_0 CMD	LINFlexD_2 RX	GTM_TOM1_IRQ7	PSI5_0 CH2 RX PSI5	ADC_SAR_1 EOC	ADC_SAR_7 EOC
40	Reserved	GTM_MCS0_IRQ3	GTM_ATOM3_IRQ1	GTM_ATOM3_IRQ3	DSPI_0 RX	LINFlexD_2 TX	GTM_TOM2_IRQ0	PSI5_0 CH2 RX SENT	ADC_SAR_2 EOC	ADC_SD_5 EOC
41	Reserved	GTM_MCS0_IRQ4	ADC_SD_2 EOC	SIUL2 REQ8	DSPI_0 TX	ADC_SAR_1 EOC	GTM_TOM2_IRQ1	ADC_SD_0 EOC	ADC_SAR_B EOC	DSPI_2 CMD
42	Reserved	GTM_MCS0_IRQ5	DSPI_1 CMD	SIUL2 REQ3	Reserved	Reserved	GTM_TOM2_IRQ2	ADC_SD_1 EOC	ADC_SD_4 EOC	DSPI_2 RX
43	Reserved	GTM_MCS0_IRQ6	DSPI_2 RX	ADC_SD_3 EOC	Reserved	Reserved	GTM_TOM2_IRQ3	ADC_SD_2 EOC	DSPI_4 CMD	DSPI_2 TX
44	Reserved	GTM_MCS0_IRQ7	DSPI_2 TX	ADC_SAR_6 EOC	Reserved	Reserved	GTM_TOM2_IRQ4	ADC_SD_3 EOC	DSPI_4 RX	SIUL2 REQ11
45	Reserved	LINFlexD_15 RX	LINFlexD_2 RX	DSPI_2 CMD	Reserved	Reserved	GTM_TOM2_IRQ5	DSPI_3 CMD	DSPI_4 TX	Reserved
46	Reserved	LINFlexD_15 TX	LINFlexD_2 TX	DSPI_1 RX	Reserved	Reserved	GTM_TOM2_IRQ6	DSPI_3 RX	Reserved	Reserved
47	Reserved	DSPI_5 RX	GTM_SPE0	DSPI_1 TX	Reserved	Reserved	GTM_TOM2_IRQ7	DSPI_3 TX	Reserved	Reserved
48	Reserved	DSPI_5 TX	GTM_SPE1	ADC_SAR_2 EOC	Reserved	Reserved	Reserved	DSPI_5 CMD	Reserved	Reserved
49	Reserved	DSPI_5 CMD	PSI5-SPS	ADC_SD_2 EOC	Reserved	Reserved	Reserved	DSPI_5 RX	Reserved	Reserved
50	Reserved	DSPI_12 CMD	PSI5-SRX	PSI5-STX	Reserved	Reserved	Reserved	DSPI_5 TX	Reserved	Reserved
51	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
52	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Table continues on the next page...

Table 7-26. DMAMUX peripheral DMA request to input source mapping (continued)

SOURCE	Peripheral DMA requests									
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3	DMAMUX_4	DMAMUX_5	DMAMUX_6	DMAMUX_7	DMAMUX_8	DMAMUX_9
53	Reserved	DSPI_0 RX	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
54	Reserved	DSPI_0 TX	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
55	Reserved	ADC_SAR_0 EOC	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
56	Reserved	ADC_SAR_4 EOC	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
57	Reserved	ADC_SD_3 EOC	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
58	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
59	Reserved	Always on	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
60	Reserved	Always on	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
61	Reserved	Always on	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
62	Reserved	Always on	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
63	Always on	Always on	Always on	Always on	Always on	Always on	Always on	Always on	Always on	Always on

Note

Just as multiple CHCONFIG registers within the same instance must not select the same source value, CHCONFIG registers among multiple instances must not select the same source value.

7.5.4 DMAMUX Memory map

The 10 instances of DMAMUX (0–9) on MPC5777M. Each instance is assigned to its own peripheral bridge slot and is allocated 512 bytes of memory space. The DMAMUX address offsets are shown in Table 7-27.

Table 7-27. DMAMUX memory map

Offset[1]	Register[3]	Access
0x0000	DMAMUX_0 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0200	DMAMUX_1 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0400	DMAMUX_2 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0600	DMAMUX_3 CHCONFIG0–CHCONFIG7 — Channel configuration registers	R/W
0x0800	DMAMUX_4 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W

Table continues on the next page...

Table 7-27. DMAMUX memory map (continued)

Offset[1]	Register[3]	Access
0x0A00	DMAMUX_5 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W
0x0C00	DMAMUX_6 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W
0x0E00	DMAMUX_7 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W
0x1000	DMAMUX_8 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W
0x1200	DMAMUX_9 CHCONFIG0–CHCONFIG15 — Channel configuration registers	R/W

1. Offsets are from DMAMUX base address (0xFFF6_C000), shown in .
2. Offsets are from DMAMUX base address (0xFFF6_C000), shown in .
3. Each channel must be configured separately via settings in a dedicated configuration channel. There is one configuration register per channel for each DMAMUX. See the DMACHMUX chapter for details.

Table 7-28. Reference links to related information

Topic	Related module	Reference
Full description	DMA Channel Multiplexer	
DMA controller description	DMA Controller	
System memory map	—	

7.6 Clocking

See Clocking for a description of the architecture for the system level clocks including clock generation, clock sources, and peripheral clocks. The chapter also provides configuration details for the Clock Monitor Unit (CMU).

7.7 Memories and memory interfaces

See the chapters shown in Table 7-29 for detailed information on memories and memory interfaces.

Table 7-29. Reference links to related information

Topic	Related module	Reference
Embedded memories	—	Embedded memories
System memory map	—	Memory Map
Platform Flash	—	Flash memory controller
Platform RAM Controller	—	RAM controller (PRAMC)
Embedded flash memory	—	Embedded Flash Memory (c55fmc)

Table continues on the next page...

Table 7-29. Reference links to related information (continued)

Topic	Related module	Reference
Decorated Storage Memory Controller	—	Decorated Storage Memory Controller (DSMC)
External Bus Interface (EBI) Support	—	External Bus Interface (EBI)

7.7.1 Flash memory controller (PFLASH) configuration

The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the crossbar switch. The configuration and control functions are accessed via control registers, which are read/write accessible only in supervisor mode.

Three of the registers, PFCR1, PFCR2, and PFAPR, control the interaction of master modules with the flash array by enabling/disabling prefetch or by controlling access to the flash array on a per-master basis. Use of the fields in these registers requires knowledge of the number assigned to each master. For example, the PFAPR[M2AP] field controls flash array access by master 2, the PFAPR[M3AP] field controls flash array access by master 3, and so on. See Table 7-7 in the crossbar switch configuration section for details. See the PFCR3 register details in the flash memory controller chapter for more details on configuring the flash controller.

Table 7-30. Reference links to related information

Topic	Related module	Reference
Full description	Flash Memory Controller (PFLASH)	Flash memory controller
Flash memory architecture overview	Flash memory, Flash controller	Embedded flash memory
Flash memory	Flash memory	Embedded Flash Memory (c55fmc)
Flash port assignments	Crossbar switch	Crossbar switch configuration
System memory map	—	Memory Map

The master assignments are shown in Table 7-31. Note that these are the same master assignments used elsewhere in the chip, for example, the SMPU.

Table 7-31. Flash controller bus master assignments

Bus Master Number	Bus Master
0	Core_0
1	Core_1
2	Core_2
3	DMA_0

Table continues on the next page...

Table 7-31. Flash controller bus master assignments (continued)

Bus Master Number	Bus Master
4	Ethernet
5	FlexRay_0
6	SIPI_1
7	SIPI_0/LFAST
8	Core_0 Debug
9	Core_1 Debug
10	Core_2 Debug
11	DMA_1
12	Reserved
13	FlexRay_1
14	HSM
15	NAR

7.7.1.1 Calibration remapping support

The PFLASH_C55FM flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM or on-chip system RAM that can be used as overlay RAM. The relationship between flash access latency and remapped overlay RAM latency is contingent upon operating frequency and intrinsic memory access times. describes the intrinsic cycle latency on a random, initial access for each supported overlay RAM target.

(See PFlash calibration remap support for more details on the relationship between flash access latency and overlay RAM access latency.

Note

Cycle times measured based on the PFLASH_C55FM clock.
See Clocking for more details on clocking.

Table 7-32. Calibration memory access times

Calibration memory	Burst access time (cycles)
On-chip overlay RAM	5
Buddy device overlay RAM	8
System RAM	9

7.7.1.2 Wait state settings for overlay flash latency matching

A remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time for a flash access as well as the required access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. The table below describes the minimum supported flash wait state setting to guarantee a remapped access matches the response latency of a flash access.

Table 7-33. Minimum wait state settings for overlay flash latency match

Overlay Target	PFCR1[RWSC] Minimum Value
BD RAM	6
On-Chip Overlay RAM	3
System RAM	6

If PFCR1[RWSC] is set sufficiently high such that the remapped burst access completes before flash read data is expected to be returned, the flash controller stalls the return of overlay data the appropriate number of additional cycles to ensure the access time matches the response latency of a flash access. Once the overlay access is stored in the flash controller mini-cache, subsequent requests that hit in the mini-cache are returned with zero wait state latency.

7.7.2 Decorated Storage Memory Controller (DSMC)

The DSMC supports five store and three load operations, including:

- Bit field inserts
- Compare-and-store
- Bitwise AND, OR, and XOR operators
- Simple memory load
- Swap
- Load-and-set-1(bit)

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

The module includes error checking logic that validates the decoration field, making sure that it is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated storage memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

The DSMC is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

For the MPC5777M device, the DSMC is instantiated six times, three times in the computational platform shell and three times in the peripheral platform shell. The access size is forced to 64-bit for system RAM, 32-bit for the core xMEMs and PBRIDGE.

Table 7-34. MPC5777M DSMC instantiations

	DSMC and memory controller association
Computational shell	System RAM
	Main Core 0 xMEM (Backdoor Port to the core local Instruction and Data Memories)[1]
	Main Core 1 xMEM (Backdoor Port to the core local Instruction and Data Memories)[1]
Peripheral shell	IOP xMEM (Backdoor Port to the core local Instruction and Data Memories)[1]
	PBRIDGE 0 (Bridge to peripheral subsystem)
	PBRIDGE 1 (Bridge to peripheral subsystem)

1. Instances of the DSMC associated with processor core tightly-coupled memory do not protect both ports—only the backdoor port used by other cores or peripherals is protected. Any IMEM or DMEM location expected to be referenced with atomic access should only be accessed by the local processor core via a decorated storage instruction.

CAUTION

Do NOT perform decorated accesses to any address spaces other than those explicitly listed above. The EBI, for example, has no knowledge of or support for decorated accesses, and a decorated access to it could result in the data at that address being overwritten with zeros.

7.7.3 External Bus Interface (EBI) Support

An EBI is accessible from the device platform with internal connection architecture that allows versatile operation supporting multiple use cases.

- EBI accesses can optionally be routed via the platform flash controller, allowing them to benefit from read buffering and line prefetch support.
  - EBI access routing is determined by dual address mapping of EBI.
  - Optimizes performance of systems that use EBI-connected external flash memories to store program code.
- EBI accesses can be optionally routed via a dedicated AXBS slave port, allowing EBI accesses to be processed simultaneously with accesses to on chip memory resources.
- Internal overlay RAMs can be remapped over the EBI for calibration purposes.
  - Enables calibration of systems that use EBI-connected external flash memories to store constant data tables.
- External EBI connected overlay RAM can be remapped over internal flash memory.
  - Enables increase in overlay RAM resource, particularly for non ED devices

7.7.3.1 e2eECC and EBI Accesses

Since the EBI does not support transfer of any checkbit data outside of the device, all data transfers involving the EBI perform a different (effectively "reversed" compared to the operations of the initiating bus master) set of ECC calculations.

Specifically, EBI write data transfers perform an ECC decode and syndrome calculation, performing any needed data corrections before the data is driven onto the EBI for the actual memory device write. On EBI read data transfers, the data from the external memory device is passed through ECC encode logic to generate the appropriate read checkbits (including the address ECC) for the transmission through the system interconnect back to the requesting bus master. For additional details refer to End-to-end Error Correction Code (e2eECC) section.

For specific details on EBI including a block diagram, signal descriptions, supported timings, memory devices, and register definitions, see the External Bus Interface (EBI).

Table 7-35. Reference links to related information

Topic	Related module	Reference
Full description	EBI	External Bus Interface (EBI)
System memory map	—	Memory Map
Clocking	—	Clocking
EBI pinout	—	Signal Description
Flash Memory Controller	Flash Memory Controller	Flash memory controller
EBI calibration remap support	Flash Memory Controller	PFlash calibration remap support

7.8 Analog modules

This section highlights the following analog modules that are implemented on MPC5777M:

Table 7-36. Analog modules

Module or Subsystem	Device-specific information	Block details
Temperature sensor	Temperature sensor configuration	Temperature Sensor Power Management Controller digital interface (PMC_dig) and Temperature Sensor interface logic
ADC subsystem	ADC overview	See individual modules.
Sigma-Delta Analog-to-Digital Converters (SDADCs)	Sigma-Delta Analog-to-Digital Converter (SDADC)	Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	Successive Approximation Register Analog-to-Digital Converter (SARADC)	Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

7.8.1 Temperature sensor configuration

The temperature sensor is used to measure the temperature of the MPC5777M device. In general, the temperature sensor generates an analog voltage that is linearly proportional to temperature. The temperature sensor also generates three digital outputs (flags) that signal over/under-temperature thresholds.

During production testing of the device, the output of the temperature sensor is sampled by SARB input channel 120 of the onboard ADC module at a predefined high temperature and also at a predefined low temperature. These two temperature sensor readings are used to calculate two constants, TSCA and TSCB, according to the formulas in Temperature Sensor.

Once calculated, the values for TSCA and TSCB are programmed during production test into the UTEST flash memory at locations 0x0040_0000 and 0x0040_0002, respectively.

The ADC bandgap voltage, used in the temperature calculation, is sampled by SARB input channel 121 of the onboard ADC module.

Software must perform the following steps in order to calculate device temperature.

1. Measure the bandgap voltage, $V_{BG}(T)$, using SARB input channel 121 of the onboard ADC module.
2. Calculate $V_{BG\_CODE}(T)$ according to the formula in Temperature Sensor.
3. Measure the temperature sensor output voltage, $V_{TSENS}(T)$, using SARB input channel 120 of the onboard ADC module.
4. Calculate $T_{TSENS\_CODE}(T)$ according to the formula in Temperature Sensor.
5. Retrieve parameters TSCA and TSCB from flash memory.
6. Calculate T according to the formula in Temperature Sensor.

Additional details are found in Power Management Controller digital interface (PMC_dig) (see Temperature Sensor interface logic).

7.8.2 Analog to Digital Converters (ADC) configuration

See Analog-to-Digital Converters (ADC) Configuration, for an overview of the ADC subsystem, including a block diagram and a summary of analog input pin multiplexing. The chapter also provides configuration details for the SARADCs and SDADCs.

7.8.2.1 SDADC configuration

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides the accurate conversion data for a wide range of applications.

MPC5777M includes 10 independent 16-bit SDADCs.

For details, see Analog-to-Digital Converters (ADC) Configuration.

7.8.2.2 SARADC configuration

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for an application. The SARADC provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has a corresponding digital interface.

The SARADCs are referred to SARADC_B, with a maximum 84:1 input mux, and different instances of Fast SARADCs, with a maximum 8:1 input mux for each. MPC5777M includes 12 independent 12-bit SARADCs (SARADC_0–SARADC_10 and SARADC_B). SARADC_B has access to all external analog input pins, access to the external multiplexed inputs, and has special safety features.

For details, see Analog-to-Digital Converters (ADC) Configuration.

7.9 Timers

Three types of timers and a complex timer subsystem are implemented on MPC5777M:

Table 7-37. Timers

Timer	Device-specific information	Block details
System Timer Module (STM)	System Timer Module (STM) configuration	System Timer Module (STM)
Software Watchdog Timer (SWT)	Software Watchdog Timer (SWT) configuration	Software Watchdog Timer (SWT)
Periodic Interrupt Timer (PIT)	PIT configuration	Periodic Interrupt Timer (PIT)
GTM104 Wrapper Module Configuration	Generic Timer Module 104 (GTM104) configuration	GTM Integration (GTMINT) Module

Each core will have a dedicated System Timer Module (STM) and a dedicated Software Watchdog Timer (SWT). The Checker core does not have a dedicated STM nor SWT.

An extra SWT (SWT_3) is available on the Peripheral core to enable a secure watchdog function.

Two Periodic Interrupt Timer (PIT) module instances are implemented. One PIT module is for 64-bit timer implementation by way of two 32-bit timer channels (0 and 1) with chaining mode; the timer value is unchanged by a non-destructive reset. The other PIT module is for general use and has eight 32-bit timer channels with no chaining mode.

The GTM104 is an implementation of the Robert Bosch GmbH GTM timer subsystem. It is not a single module; it is a complex timer subsystem that consists of many modules. It can be used to implement highly complex timer functions.

7.9.1 System Timer Module (STM) configuration

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the appropriate system clock divided by an 8-bit prescale value (1 to 256).

MPC5777M includes three STMs, one for each core, with four 32-bit compare channels in each STM.

Table 7-38. System Timer Module (STM) instances

Instance	Description	Number of Channels	Number of Registers (32-bit)
STM0	Intended for Main Core_0 in the high-speed computational clock domain.	4	14
STM1	Intended for Main Core_1 in the high-speed computational clock domain.	4	14
STM2	Intended for Peripheral Core_2 in the peripheral clock domain.	4	14

7.9.2 Software Watchdog Timer (SWT) configuration

MPC5777M includes four SWTs.

Table 7-39. Software Watchdog Timer (SWT) instances

Instance	Description
SWT0	Intended for Main Core_0 in the high-speed computational clock domain.
SWT1	Intended for Main Core_1 in the high-speed computational clock domain.
SWT2	Intended for Peripheral Core_2 in the peripheral clock domain. Elapsed times recorded in timeout register are different from SWT0 and SWT1.
SWT3[1]	Intended as security watchdog timer in the peripheral clock domain. Elapsed times recorded in timeout registers are different from SWT0 and SWT1

1. SWT3 can be stopped only when:
    1. all cores are either stopped or disabled, AND
    2. at least 1 core is enabled (note that this core is also stopped, to meet 1).

The SWT has the following features:

- 32-bit time-out register to set the time-out period

- Internal 16 MHz RC oscillator clock that is used for timer operation
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits

7.9.2.1 Security watchdog

SWT (SWT3 Security) is identical to the other SWT modules except the Instruction Address Compare feature is NOT instantiated.It is anticipated that each CPU would use an SWT to ensure software tasks continue to execute correctly.

If a CPU malfunctions (either software or hardware) the SWT is no longer serviced and a reset occurs. This is referred to as a Task Watchdog. The Security Watchdog is slightly different from the Task Watchdog.

The Security watchdog is intended to be serviced when the CPU executes an instruction at a specific address location. This address location would be an address within some fundamental section of security code. If this address location is not executed regularly, it implies that a malicious security attach has occurred and the device should be reset.

Additionally, the Security watchdog is deactivated by a device Power-On-Reset.

This security code could be executed on any of the CPUs. The SWT for that CPU would perform the Security Watchdog (SWT0, SWT1 or SWT2) because it includes the IAC feature. The Task Watchdog for that CPU would need to use SWT3 (Security), which does NOT include the IAC feature.

As a result the SWT3 (Security) is never used to perform the Security Watchdog; but is used for Task Watchdog to replace the SWT that IS being used for Security Watchdog

7.9.2.2 Reset assertion

The SWT can assert a reset when the watchdog timer expires. This reset causes a system reset equivalent to assertion of the RESET pin.

7.9.2.3 Default configuration

On MPC5777M, the SWTs come out of reset with the following default values. Table 7-40 summarizes the reset values. Minimum timeout value is 256 cycles for SWT0, SWT1, and SWT2.

Table 7-40. Register reset values

SWT instance	Register	Reset value
SWT0	SWT Control Register (SWT_CR)	0xFF00_010A
SWT0	SWT Time-out (SWT_TO)	0x0005_FCD0
SWT1	SWT Control Register (SWT_CR)	0xFF00_010A
SWT1	SWT Time-out (SWT_TO)	0x0005_FCD0
SWT2	SWT Control Register (SWT_CR)[1]	0xFF00_011B
SWT2	SWT Time-out (SWT_TO)	0x0003_FDE0
SWT3	SWT Control Register (SWT_CR)	0xFF00_010A
SWT3	SWT Time-out (SWT_TO)	0x0005_FCD0

1. The SWT can be disabled by accepting a DCI-driven signal to clear the SWT_CR[WEN] bit. For SWT2, the default reset value for that bit is '1', but the DCI can override that default value by setting it to '0'. If the signal is de-asserted, there is no change to the SWT_CR[WEN] value. (The DCI can only disable the SWT, it cannot enable it.) See the DCI chapter for details on the system watchdog control.

The Security Watchdog (SWT3) is enabled using DCF Records in conjunction with the BAF. Please see Security watchdog configuration for details.

For SWT[0,1,3], the SWT_CR register reset value of 0xFF00_011B selects:

- All masters allowed access
- Reset on invalid access
- Oscillator clock selected
- Counter stops in debug mode
- Watchdog is disabled

For SWT[2], the SWT_CR register reset value of 0xFF00_010A selects:

- All masters allowed access
- Reset on invalid access
- Oscillator clock selected
- Counter stops in debug mode
- Soft locked
- Watchdog is enabled

7.9.2.4 Service mode watchpoint input

Each SWT comes out of reset in 'Fixed Service Sequence' mode (SWT_CR[SMD] = 2’b00), but the Instruction Address Compare 8 (IAC8) register (SPR 568) can be used for servicing by using the SWT jd_watchpt input.

To use SWT address execution modes for this purpose, the e200z4 core IAC8 register signal jd_watchpt[15] is connected to the SWT jd_watchpt input. (See the "Watchpoint support" section for details.) The jd_watchpt[15] signal tells the SWT that an IAC8 compare occurred.

Set SWT_CR[SMD] to one of the following so the SWT is serviced by executing code at the address loaded into the IAC8 register:

- 2’b10 for Fixed Address Execution; the IAC8 register cannot be updated while the watchdog is enabled. If the SWT is in 'Fixed Address Execution' mode, an SWT output signal tells the core; this ‘freezes’ changes to the IAC8 register when the SWT is enabled.
- 2’b11 for Incremental Address Execution; the IAC8 register can be updated.

7.9.3 PIT configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

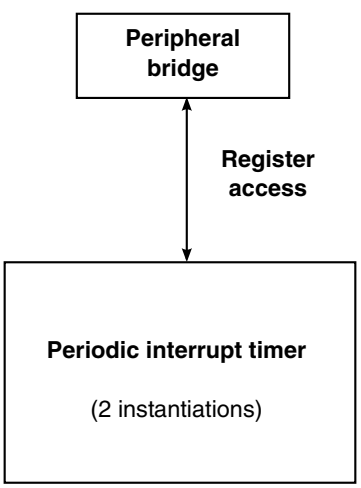


Figure 7-17. PIT configuration

Table 7-41. Reference links to related information

Topic	Related module	Reference
Full description	PIT	PIT chapter

Table continues on the next page...

Table 7-41. Reference links to related information (continued)

Topic	Related module	Reference
System memory map	-	Memory Map chapter
Clocking	-	Clocking chapter
Power management	-	Power Management chapter
DMA channel assignments	Direct memory access multiplexer	DMAMUX chapter

7.9.3.1 PIT Instantiation

Table 7-42. PIT Information

PIT Module	No of Channels	Description
PIT0	8 + RTI	PIT0 has 8 standard channels and 1 RTI channel. In PIT0, all registers are cleared by any reset.
PIT1	2	PIT1 has chaining mode to implement a 64-bit timer and the timer value is unchanged by a functional reset.

7.9.3.2 PIT/DMA Periodic Trigger Assignments

In PIT1, DMA is not supported. The PIT0 generates periodic trigger events to the DMA Mux as shown in the table below.

Table 7-43. PIT0 channel assignments for periodic DMA triggering

DMA Channel	PIT0 Channel
DMA Channel 8	PIT0 Channel 0
DMA Channel 9	PIT0 Channel 1
DMA Channel 10	PIT0 Channel 2
DMA Channel 11	PIT0 Channel 3
DMA Channel 12	PIT0 Channel 4
DMA Channel 16	PIT0 Channel 5
DMA Channel 32	PIT0 Channel 6
DMA Channel 48	PIT0 Channel 7

7.9.3.3 PIT Registers

The following tables show the implementation of PIT registers in this MCU.

Table 7-44. PIT0 memory map

Absolute address (hex)	Register name
FFF8_4000	PIT Module Control Register (PIT0_MCR)[1]
FFF8_40E0 - FFF8_40E4	Reserved
FFF8_40F0 - FFF8_40FC	RTI Timer Channel Registers (PIT0_RTI_LDVAL, PIT0_RTI_CVAL, PIT0_RTI_TCTRL, PIT0_RTI_TFLG)
FFF8_4100 - FFF8_417C	Timer Channel [0:7] Registers (PIT0_LDVALn, PIT0_CVALn, PIT0_TCTRLn, PIT0_TFLGn)
FFF8_4180 - FFF8_7FFF	Reserved

1. The reset value of MCR is 0x06.

Table 7-45. PIT1 memory map

Absolute address (hex)	Register name
FFF8_0000	PIT Module Control Register (PIT1_MCR)[1]
FFF8_00E0	PIT Upper Lifetime Timer Register (PIT1_LTMR64H)
FFF8_00E4	PIT Lower Lifetime Timer Register (PIT1_LTMR64L)
FFF8_00F0 - FFF8_00F8	Reserved
FFF8_0100 - FFF8_011C	Timer Channel [0:1] Registers (PIT1_LDVALn, PIT1_CVALn, PIT1_TCTRLn, PIT1_TFLGn)
FFF8_0120 - FFF8_3FFF	Reserved

1. The reset value of MCR is 0x02.

7.9.3.4 PIT clocking

The PIT0 standard channels are clocked with the peripheral clock (PER_CLK).

The PIT0 RTI channel clock source is selectable between XOSC and RCOSC via the auxiliary clock selector 9.

The PIT 1 channels are clocked with the peripheral clock (PER_CLK).

7.9.4 GTM subsystem configuration

7.9.4.1 GTM subsystem

This diagram illustrates the parts of the GTM subsystem.

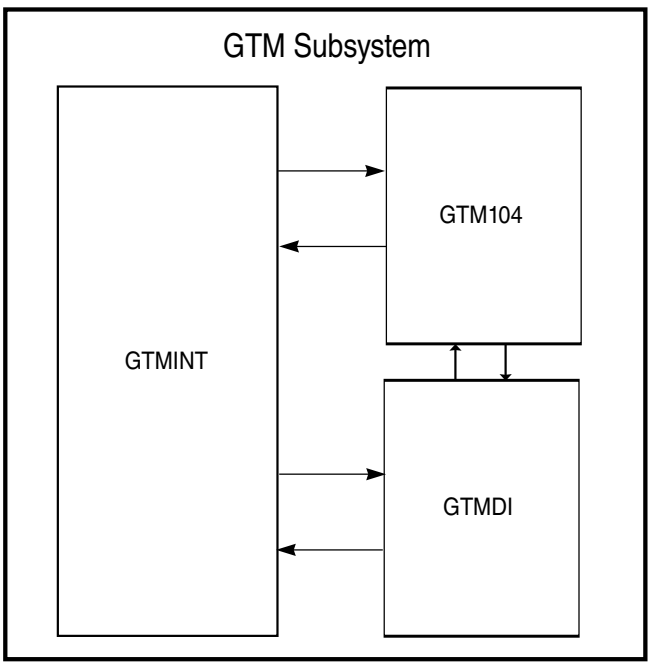


Figure 7-18. GTM subsystem

7.9.4.2 Parts of the GTM subsystem

This table describes the parts of the GTM subsystem.

Part	Function
GTMINT	Provides reset control, clock control, error-correcting code (ECC) functionality, interrupt and DMA control, and access to GTM104's memory-mapped registers. For debug accesses from GTMDI to GTM104 registers, prevents destructive read accesses into the GTM104 registers. For more information, see GTM Integration (GTMINT) Module.
GTM104	Provides programmable timer functionality. For more information, see the see Generic Timer Module (GTM104) chapter.
GTMDI	Provides real-time development capabilities for the GTM subsystem. For more information, see Development Interface (GTMDI) Module.

7.9.4.3 GTM subsystem clock sources

The system clock and the peripheral clocks to the GTM subsystem are provided by the Clock Generation Module (MC_CGM).

GTM subsystem clock	MC_CGM source	Maximum frequency
SYS_CLK	SXBAR_CLK	100 MHz
GTMIP_CLK	PER_CLK	80 MHz
GTMDI_CLK	PER_CLK	80 MHz
GTMINT_CLK	PER_CLK	80 MHz

7.9.4.4 Reset value of GTM Module Configuration Register (GTMMCR)

GTMMCR contains the control bits to configure the general operation of GTMINT and GTM-IP. The reset value of the register is 4000_4000h, which means that after reset the GTM subsystem is in Stop mode.

7.10 Communication interfaces

7.10.1 FEC interface selection

On this chip, the FEC interface depends on the settings of the RCR (in the FEC) and the UTEST Miscellaneous DCF Client, as shown in Table 7-46.

Table 7-46. FEC interface selection_a

RCR[MII_MODE]	"RMII MODE" field of the UTEST Miscellaneous DCF Client	Interface used by the FEC
0	0 or 1	7-wire
1	0	MII
	1	RMII

Refer to Fast Ethernet Controller (FEC) for detailed description.

7.10.2 CAN subsystem configuration

7.10.2.1 CAN nodes

Table 7-47 lists the number of CAN nodes and features available for the specific CAN nodes.

Table 7-47. CAN bus controllers

Types	Instances	CAN FD support	Debug on CAN Support
M_CAN	M_CAN_1	Yes	Yes
	M_CAN_2	Yes	Yes
	M_CAN_3	Yes	No
	M_CAN_4	Yes	No
M_CAN w ith TTCAN	M_TTCAN_0	Yes	No

7.10.3 DSPI configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

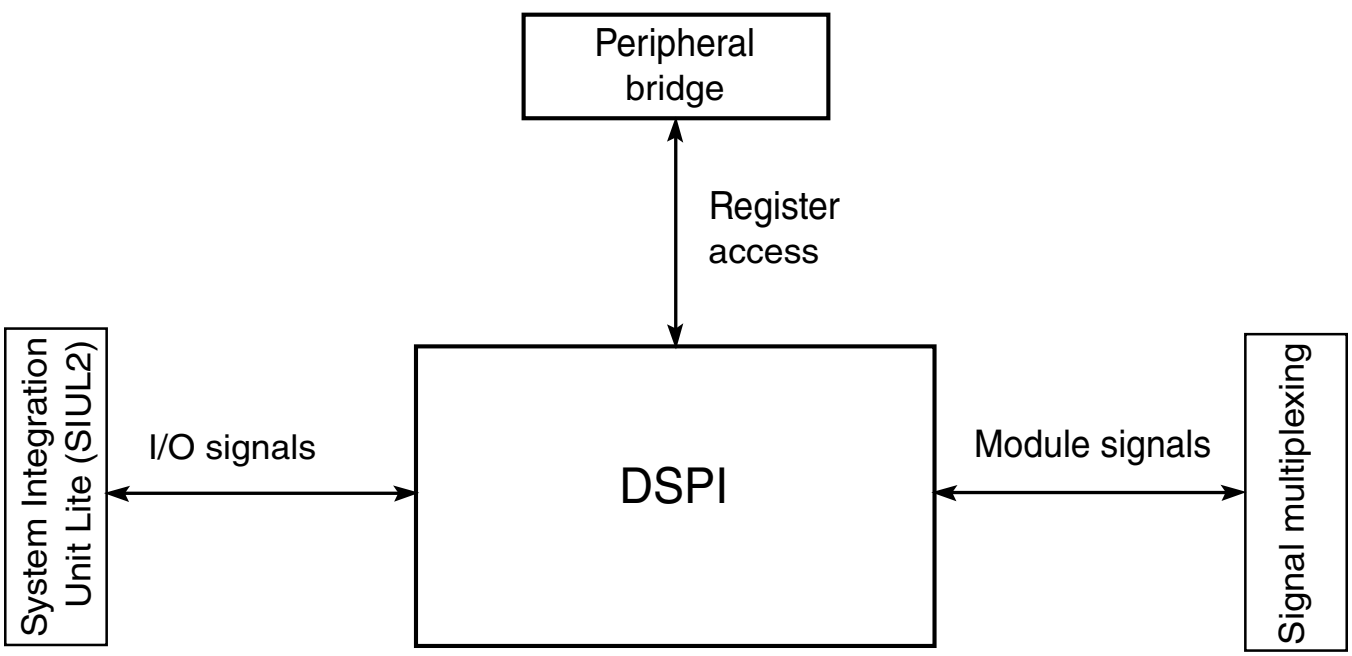


Figure 7-19. DSPI configuration

Table 7-48. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	Deserial Serial Peripheral Interface (DSPI)
System memory map	—	Memory Map
Clocking	—	Clocking
Signal Multiplexing	Port control	System Integration Unit Lite2 (SIUL2)
System Module	System Integration Unit Lite	System Integration Unit Lite2 (SIUL2)
DMA Channel Assignments	Direct memory access multiplexer	Direct Memory Access Multiplexer (DMAMUX)

7.10.3.1 DSPI instantiation

This device contains eight DSPI modules. Table 7-49 shows the instantiation information for each module.

Table 7-49. DSPI instantiation

DSPI Module	CTARs	TX FIFO Depth	RX FIFO Depth	CMD FIFO Depth	PCS Signals	LVDS Support	DataSerialization Support
DSPI0	8	4	4	4	PCS[7:0]	Not Available	—
DSPI1	8	4	4	4	PCS[7:0]	Not Available	—
DSPI2	8	4	4	4	PCS[7], PCS[4:0]	Available[1]	—
DSPI3	8	4	4	PCS[7:0]	Not Available	—	

Table continues on the next page...

Table 7-49. DSPI instantiation (continued)

DSPI Module	CTARs	TX FIFO Depth	RX FIFO Depth	CMD FIFO Depth	PCS Signals	LVDS Support	Data Serialization Support
DSPI4[2]	8	4	4	4	PCS[7:4], PCS[2:0]	Available[3]	32
DSPI5	8	4	4	4	PCS[7:0]	Available[4]	32
DSPI6	8	4	4	4	PCS[5:4], PCS[2:0]	Available	32
DSPI12	8	4	4	4	PCS[2:0]	Not Available	—

1. Does not support Microsecond Bus.
2. Available in half duplex mode.
3. Supports Microsecond Bus SCK and SOUT LVDS signal pairs.
4. Supports Microsecond Bus SCK and SOUT LVDS signal pairs; supports Differential DSPI with SCK, SIN and SOUT LVDS signal pairs.

7.10.3.2 DSPI half duplex operation

Half duplex operation is not supported within the DSPI module, but it is configured by connecting the SIN and SOUT together within the SIUL2 and operating the combined pad in open drain mode. Software must ensure that the SOUT signal is passive (high) during SIN reception. This can be done by reconfiguring the SIUL2 to disconnect the SOUT or by writing a data frame that is all '1's.

7.10.3.3 Serialized timed I/O

The timer sources and injection pin inputs to be DSPI serialized downstream outputs are selected though the SIUL2 and selected through the MSCR registers. Inversion of these signals is also handled in the SIUL2. Refer to System Integration Unit Lite2 (SIUL2), for details.

7.10.3.4 High Speed and MicroSecond Channel (MSC) Support

The baud rate generator in the DSPI is clocked from the peripheral clock. The DSPI module can operate at a baud rate of up to half the peripheral clock. The peripheral clock frequency specification is 16 MHz to a maximum of 100 MHz. As long as the peripheral clock is 100 MHz, 50 MHz baud rate is achievable.

To support microsecond channel functionality, pairs of LVDS signals are implemented for the appropriate DSPI signals. Therefore, SOUT4, SOUT_5, SOUT_6, SCK_4, SCK_5, SCK_6, have LDVS capability. In addition, SIUL2 MSCR muxing is implemented to mux the SOUT and CLK onto these LVDS signals well as standard CMOS signals. Finally, in the SIUL2 MSCR pin muxing, 4 input pins can be routed to 32 possible serialization inputs, each selectable for a specific DSPI.

The following four external pins can be inserted into the MSC data stream:

- 0000_0010 IO_PAD PC[2]
- 0000_0011 IO_PAD PD[8]
- 0000_0100 IO_PAD PF[3]
- 0000_0101 IO_PAD PJ[4]

Details of these pin assignments can be found in the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the Input Multiplexing Table tab.

7.10.3.5 DSPI HT external trigger

External hardware triggers are not supported.

7.10.3.6 DSPI ITSB mode

For the ITSB trigger, the internal DSPI trigger (TRGPRD in DSICR1) is used. Use of external trigger is not supported.

7.10.3.7 DSPI registers

The following table shows the absolute addresses of DSPI module registers in this MCU.

Table 7-50. DSPI registers memory map

Absolute address (hex)	Module
FBE7_0000	DSPI2 registers
FBE7_4000	DSPI3 registers
FBE7_8000	DSPI5 registers
FFE7_0000	DSPI0 registers
FFE7_4000	DSPI1 registers
FFE7_8000	DSPI4 registers
FFE7_C000	DSPI6 registers
FFE8_8000	DSPI12 registers

7.10.3.7.1 DSPIx_MCR implementation

The following figure shows the implementation of DSPIx_MCR register in this MCU.

Address Base + 0x0000 Access: User read/write

:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	PCSIS[7:0]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DOZE[1]	MDIS	DIS_TXF	DIS_RXF	0	0	SMPL_PT		0	0	0	0	XSPI	FCPCS	PES	HALT
W					CLR_TXF	CLR_RXF										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 7-20. DSPI Module Configuration Register (DSPIx_MCR)

7.10.4 FlexRay Configuration

The FlexRay communications system is designed to provide high-speed deterministic distributed control for advanced automotive applications. Its dual-channel architecture supporting data rates up to 10 Mbit/s per channel offers system-wide redundancy that supports the reliability requirements of safety system.

For details, see the FlexRay Communication Controller (FlexRay).

7.10.4.1 Number of FlexRay modules

This device has two FlexRay modules.

7.10.4.2 FlexRay signals pin assignment

This table lists the pin names for the FlexRay signals that are connected to pins on the boundary of the chip.

Table 7-51. Flexray_0 pin names

Signal description	Pin name
FR_A_RX (Receive Data Channel A)	RXD0_A
FR_A_TX (Transmit Data Channel A)	TXD0_A
FR_A_TX_EN (Transmit Enable Channel A)	TXEN0_A
FR_B_RX (Receive Data Channel B)	RXD0_B
FR_B_TX (Transmit Data Channel B)	TXD0_B
FR_B_TX_EN (Transmit Enable Channel B)	TXEN0_B
FR_DBG[0] (Debug Strobe Signal 0)	FR0_DBG0
FR_DBG[1] (Debug Strobe Signal 1)	FR0_DBG1
FR_DBG[2] (Debug Strobe Signal 2)	FR0_DBG2
FR_DBG[3] (Debug Strobe Signal 3)	FR0_DBG3

Table 7-52. Flexray_1 pin names

Signal description	Pin name
FR_A_RX (Receive Data Channel A)	RXD1_A
FR_A_TX (Transmit Data Channel A)	TXD1_A
FR_A_TX_EN (Transmit Enable Channel A)	TXEN1_A
FR_B_RX (Receive Data Channel B)	RXD1_B
FR_B_TX (Transmit Data Channel B)	TXD1_B

Table continues on the next page...

Table 7-52. Flexray_1 pin names (continued)

Signal description	Pin name
FR_B_TX_EN (Transmit Enable Channel B)	TXEN1_B
FR_DBG[0] (Debug Strobe Signal 0)	FR1_DBG0
FR_DBG[1] (Debug Strobe Signal 1)	FR1_DBG1
FR_DBG[2] (Debug Strobe Signal 2)	FR1_DBG2
FR_DBG[3] (Debug Strobe Signal 3)	FR1_DBG3

7.10.4.3 Stopwatch timer input

Stopwatch timer input is routed to FlexRay module (selected by SIUL MSCR register) via:

- STM0_CMP0
- STM1_CMP0
- STM2_CMP0
- PIT0_CH2

Details of these pin assignments can be found in the MPC5777M I/O Signal Description and Input Multiplexing Tables (excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the excel file to open it and select the Input Multiplexing Table tab.

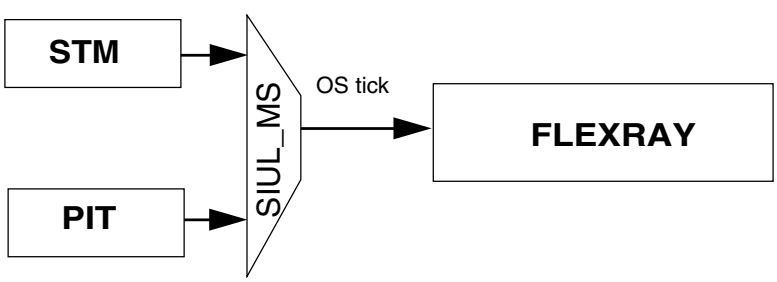


Figure 7-21. Stopwatch timer source muxing

7.10.5 Inter-Integrated Circuit (I2C) Configuration

MPC5777M has two I^2C modules: IIC_0 and IIC_1. See Inter-Integrated Circuit (I^2C) for detailed information.

7.10.6 PSI5 Controller Configuration

The PSI5 chapter has 2 instances. Parameters for the 2 instances are listed below:

Table 7-53. PSI5 controller generics.

Signal name	Description	PSI5_0	PSI5_1
PSI5_NUM_CH	Number of channels	3	2
PSI5_PDIS_RST	Default State of the IP after reset	1	1

PSI5_0 has three channels (Channel0, Channel1 and Channel2). PSI5_1 has two channels (Channel0 and Channel1). PSI5_0 is a master instance here hence the PSI5_GCR register for this instance controls all the 5channels; PSI5_0 (Channel 0, Channel 1 and Channel2) and PSI5_1 (Channel0 and Channel1). PSI5_1 instance does not contain the PSI5_GCR register and hence the corresponding PSI5_GCR address location for PSI5_1 is reserved. Further, with respect to the PSI5 base address, the unimplemented reserved locations for PSI5_0 extend from 0x0544 to 0xFFFF and unimplemented reserved locations for PSI5_1 extend from 0x0388 to 0xFFFF.

7.10.7 SENT receiver (SRX) configuration

The MPC5777M has two SENT modules. See SENT Receiver (SRX) for detailed information.

7.10.7.1 SRX channels

The following table shows the SENT receiver channels for each instance configured on this chip.

Table 7-54. SENT channel configuration

Instance	Channels (CH)
SENT 0	8 Channels: SENT_0_CH0 SENT_0_CH1 SENT_0_CH2 SENT_0_CH3 SENT_0_CH4 SENT_0_CH5

Table continues on the next page...

Table 7-54. SENT channel configuration (continued)

Instance	Channels (CH)
	SENT_0_CH6
	SENT_0_CH7
SENT 1	7 Channels:
	SENT_1_CH0
	SENT_1_CH1
	SENT_1_CH2
	SENT_1_CH3
	SENT_1_CH4
	SENT_1_CH5
	SENT_1_CH6

7.10.7.2 SRX registers

The following table lists the address offset and memory map for both SENT receiver (SRX) instances for this chip.

Table 7-55. SENT Receiver memory map

Offset	Register Name
0x0000	GBL_CTRL— Global Control Register
0x0004	CHNL_EN—Channel Enable Register
0x0008	GBL_STATUS—Global Status Register
0x000C	FMSG_RDY—Fast Message Ready Register
0x0010	SMSG_RDY—Slow Serial Message Ready Register
0x0014	Reserved
0x0018	DATA_CTRL1— Data Control Register 1
0x001C	DATA_CTRL2— Data Control Register 2
0x0020	Reserved
0x0024	Reserved
0x0028	FDMA_CTRL—Fast Message DMA Control Register
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register
0x0030	Reserved
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register
0x003C	Reserved
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register

Table continues on the next page...

Table 7-55. SENT Receiver memory map (continued)

Offset	Register Name
0x004C	Reserved
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register
0x005C	Reserved
0x0060	CH0_CLK_CTRL—Channel 'n' Clock Control Register
0x0064	CH0_STATUS—Channel 'n' Status Register
0x0068	CH0_CONFIG—Channel 'n' Configuration Register
0x006C	Reserved
0x0070	CH1_CLK_CTRL—Channel 'n' Clock Control Register
0x0074	CH1_STATUS—Channel 'n' Status Register
0x0078	CH1_CONFIG—Channel 'n' Configuration Register
0x007C	Reserved
0x0080	CH2_CLK_CTRL—Channel 'n' Clock Control Register
0x0084	CH2_STATUS—Channel 'n' Status Register
0x0088	CH2_CONFIG—Channel 'n' Configuration Register
0x008C	Reserved
0x0090	CH3_CLK_CTRL—Channel 'n' Clock Control Register
0x0094	CH3_STATUS—Channel 'n' Status Register
0x0098	CH3_CONFIG—Channel 'n' Configuration Register
0x009C	Reserved
0x00A0	CH4_CLK_CTRL—Channel 'n' Clock Control Register
0x00A4	CH4_STATUS—Channel 'n' Status Register
0x00A8	CH4_CONFIG—Channel 'n' Configuration Register
0x00AC	Reserved
0x00B0	CH5_CLK_CTRL—Channel 'n' Clock Control Register
0x00B4	CH5_STATUS—Channel 'n' Status Register
0x00B8	CH5_CONFIG—Channel 'n' Configuration Register
0x00BC	Reserved
0x00C0	CH6_CLK_CTRL—Channel 'n' Clock Control Register
0x00C4	CH6_STATUS—Channel 'n' Status Register
0x00C8	CH6_CONFIG—Channel 'n' Configuration Register
0x00CC	Reserved
0x00D0	CH7_CLK_CTRL—Channel 'n' Clock Control Register
0x00D4	CH7_STATUS—Channel 'n' Status Register
0x00D8	CH7_CONFIG—Channel 'n' Configuration Register
0x00DC - 0x015F	Reserved
0x0160	CH0_FMSG_DATA—Channel 'n' Fast Message Register
0x0164	CH0_FMSG_CRC—Channel 'n' Fast Message CRC Register

Table continues on the next page...

Table 7-55. SENT Receiver memory map (continued)

Offset	Register Name
0x0168	CH0_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x016C	CH0_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x0170	CH0_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x0174	CH0_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x0178	CH1_FMSG_DATA—Channel 'n' Fast Message Register
0x017C	CH1_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x0180	CH1_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x0184	CH1_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x0188	CH1_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x018C	CH1_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x0190	CH2_FMSG_DATA—Channel 'n' Fast Message Register
0x0194	CH2_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x0198	CH2_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x019C	CH2_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01A0	CH2_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01A4	CH2_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01A8	CH3_FMSG_DATA—Channel 'n' Fast Message Register
0x01AC	CH3_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x01B0	CH3_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x01B4	CH3_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01B8	CH3_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01BC	CH3_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01C0	CH4_FMSG_DATA—Channel 'n' Fast Message Register
0x01C4	CH4_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x01C8	CH4_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x01CC	CH4_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01D0	CH4_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01D4	CH4_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01D8	CH5_FMSG_DATA—Channel 'n' Fast Message Register
0x01DC	CH5_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x01E0	CH5_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x01E4	CH5_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x01E8	CH5_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x01EC	CH5_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x01F0	CH6_FMSG_DATA—Channel 'n' Fast Message Register
0x01F4	CH6_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x01F8	CH6_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x01FC	CH6_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x0200	CH6_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)

Table continues on the next page...

Table 7-55. SENT Receiver memory map (continued)

Offset	Register Name
0x0204	CH6_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x0208	CH7_FMSG_DATA—Channel 'n' Fast Message Register
0x020C	CH7_FMSG_CRC—Channel 'n' Fast Message CRC Register
0x0210	CH7_FMSG_TS—Channel 'n' Fast Message Time Stamp Register
0x0214	CH7_SMSG_BIT3—Channel 'n' Serial Message Register (Bit 3)
0x0218	CH7_SMSG_BIT2—Channel 'n' Serial Message Register (Bit 2)
0x021C	CH7_SMSG_TS—Channel 'n' Serial Message Time Stamp Register
0x0220 - 0x02E0	Reserved

The LINFlexD has 6 instances on this chip. Configuration for the 6 instances are listed below:

Table 7-56. LINFlexD configurations

Description	linflex_0	linflex_1	linflex_2	linflex_14	linflex_15	linflex_16
Number of filters implemented	**16**	**0**	**0**	**0**	**0**	**0**
Number of Tx DMA channels	**1**	**1**	**1**	**1**	**1**	**1**
Number of Rx DMA channels	1l	1	1	1	1	1
LIN operation mode	master/slave	master	master	master	master	master
Auto synchronization	yes	no	no	no	no	no

7.11 Reset and boot modules

7.11.1 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the MPC5777M. For a comprehensive description of the SSCM, please reference the SSCM's dedicated chapter.

7.11.1.1 SSCM instantiation

There is one SSCM instance in the MPC5777M device. The SSCM is part of the reset and boot sub-system.

7.11.1.2 Device specific features

The SSCM allows Boot Assist from Flash (BAF), LINFlexD search at boot and utilizes PASS security concept. It also allows for external booting from the LINFlex port, but with no autobaud feature. The device specific resets value for the SSCM_MEMCONFIG register are shown in Table 7-57.

Table 7-57. SSCM_MEMCONFIG Reset Values

Field	Field Definition	Value
MREV	Minor Revision	0000b
JPIN	JTAG Part ID	30Eh

Table 7-58 shows the address locations that the SSCM uses to search for startup information.

Table 7-58. SSCM startup information addresses

Description	Value
The location at which the SSCM will look for the UTEST DCF records (UTEST_OFFSET).	0040_0000h
Location of the Boot Assist code in Flash (BAF)	0040_4000h
Specify the locations which are searched for a valid HSM boot header. (starting from the lowest address)	60_C000h 61_0000h 62_0000h

7.11.1.3 Protected registers

The SSCM module has a built-in register protection mechanism that affects the behavior of write operations to one or more registers. After software has activated the protection for a register, writes can only be performed in supervisor mode until the protection is either deactivated by software or the device undergoes reset.

See the Register protection section of this chapter for a list of affected registers.

See the Register protection chapter for a detailed description of the register protection mechanism.

7.11.1.4 Memory map and register description

Table 7-59. Reference text to related information

Topic	Related Module	Reference
Reset	Reset Generation Module (MC_RGM)	Reset Generation Module (MC_RGM)
Reset/Boot	Reset and Boot	Reset and Boot
Boot	Boot Assist from Flash (BAF)	Boot Assist Flash (BAF)
Register Protection	Register Protection (REG_PROT)	Register Protection (REG_PROT)
Register Protection	Register Protection Configuration	System Status and Control Module (SSCM) protected registers
Serial Boot	LINFlexD	LINFlexD
Security	PASS	Password and Device Security Module (PASS)
Security	HSM	See the *Microcontroller Security Manual*

7.11.2 Boot Assist Flash (BAF) configuration

7.11.2.1 BAF memory map

The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in Table 7-60.

Table 7-60. BAF memory map

BAF component	Address
BAF address range	0040_4000h – 0040_7FFFh
BAF version number	0040_4000h
BAF code entry point	0040_4100h
BAF code exit point	5280_0080h
BAF data area and stack	5280_0000h – 5280_01FFh

Table 7-61 shows UTEST memory addresses used by the BAF.

Table 7-61. UTEST addresses

UTEST component	Address
Soft DCF Record Start Address	0040_00C0h
DCF Start Record	0040_0200h

7.11.2.2 LINFlexD_0 and MCAN_1 pin configuration

Table 7-62 shows the device pin configurations for LINFlexD_0 and MCAN_1 modules.

Table 7-62. MCAN_1 and LINFlexD_1 pin configuration

Pads	Reset Function	Initial Serial Boot mode		Serial Boot mode after valid CAN message received	Serial Boot mode after valid LINFlexD message received
		Function	Pad configuration	Function	Function
PA[10]	GPIO Input buffer off Output buffer off	MCAN_1 Tx	Output: Push/Pull medium drive no pullup/pulldown	MCAN_1 Tx	LINFlexD_0 Tx
PA[11]	GPIO Input buffer off Output buffer off	LINFlexD_0 Rx and MCAN_1 Rx	Input: pullup hystersis TTL voltage level	MCAN_1 Rx	LINFlexD_0 Rx

7.11.2.3 Search boot header and boot options

The BAF searches the flash memory for the boot header, checking the first word location of the 4 × 16 KB blocks, then the first of the 4 × 256 KB blocks. The blocks are searched in the order shown in Table 7-63. Once a Boot Header is found, no further blocks are searched.

Table 7-63. Locations of boot headers

Search order	Block	Address
1	16KB Code Flash block 0	00FC_0000h
2	16KB Code Flash block 1	00FC_4000h
3	16KB Code Flash block 2	00FC_8000h
4	16KB Code Flash block 3	00FC_C000h
5	256KB Code Flash block 0	0100_0000h
6	256KB Code Flash block 1	0104_0000h
7	256KB Code Flash block 2	0108_0000h
8	256KB Code Flash block 3	010C_0000h

The first header found that contains the value 005Ah in the first halfword is valid for booting. The whole header structure is shown in Table 7-64.

Table 7-64. Boot header structure

Address Offset	Contents
00h	Boot Header Configuration (see Table 7-65)
04h	CPU2 Reset Vector
08h	Configuration Bits
0Ch	Configuration Bits
10h	CPU0 Reset Vector
14h	CPU1 Reset Vector
18h	CPUC Reset Vector

Table 7-65. Boot header configuration

Offset 00h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				5				A			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				CPU1	CPUC	CPU0	CPU2

Table 7-66. Boot header configuration field description

Field	Description
CPU1	CPU1 is enabled (will start execution at CPU1 Reset Vector)
CPUC	CPU_SC is enabled (will start execution at CPU_SC Reset Vector, must be the same as CPU0 Reset Vector)
CPU0	CPU0 is enabled (will start execution at CPU0 Reset Vector)
CPU2 (IOP)	CPU2 (IOP) is enabled (starts execution at CPU2 Reset Vector)

The cores that are enabled in the boot header configuration field are enabled by programming MC_ME_CCTL*n*[DRUN] register to run in all DRUN modes and their reset vector is set in register MC_ME_CADDR*n*. Then, the memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally, a mode change (from DRUN to DRUN again) is requested from the MC_ME module to activate the new core configuration and make the cores jump to their respective reset vectors. At this point the BAF has finished its operation.

7.12 Power Management modules

7.12.1 PMC ADC test mode

All of the internal Power Management Controller (PMC) system analogue signals can be monitored by one of the Analog-to-Digital Converters (ADCs). SARADC_B channel 110 is dedicated for these monitoring purposes. The PMC ADC test mode is used for calibration, diagnostics during test, or for the user to actively monitor the signal levels. PMC signals are buffered to ensure correct ADC sampling. Supply for PMC to ADC buffer and transmission gates is the main PMC supply.

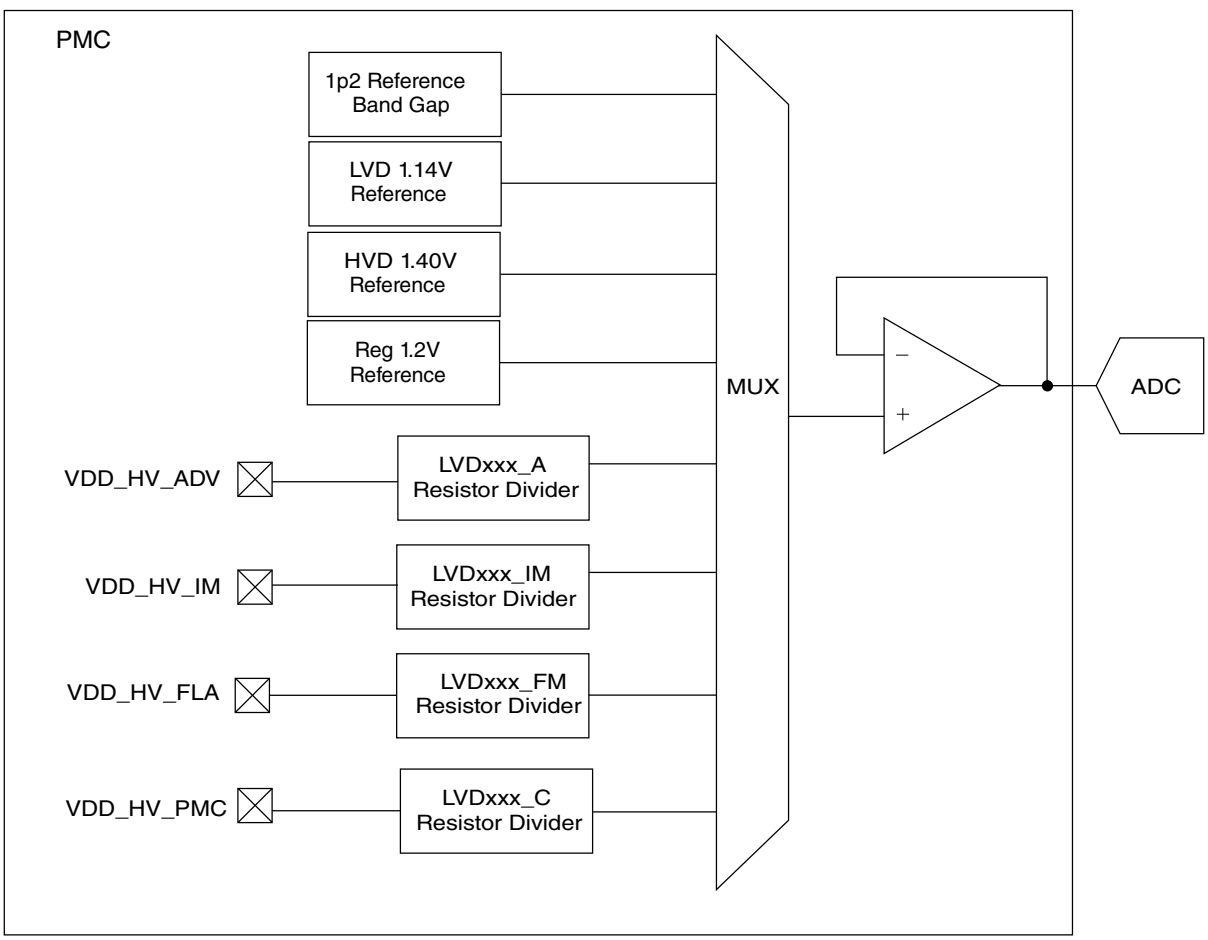


Figure 7-22. Simplified PMC ADC test multiplexer block diagram

The table below lists the parameters that can be monitored, the ADC channel select register value, and reference to the expected nominal value. Use maximum ADC sample time for most accurate results.

Table 7-67. Parameter monitoring via ADC channel

PMC ADC Mode Signal Description	ADC Channel Select Register	ADC Mode Description	Nominal value expected
Internal Voltage[1]	6b101000	Internal Voltage	0.855V
POR for security[1]	6b000001	Power-On-Reset on LV core supply for security	1.0V
POR for security[1]	6b011111	Power-On-Reset on LV flash supply for security	1.0V
LVD096_C[1]	6b011110	Power-On-Reset on LV core supply	See VLVD096 in DS
LVD096_F[1]	6b011101	Power-On-Reset on LV flash supply	See VLVD096 in DS
LVD108_B[1]	6b011100	LVD on Buddy device supply	1.0V
LVD112_C[1]	6b011011	LVD on Low voltage supply core (cold point)	See LVD112 in DS
LVD108_F[1]	6b011010	LVD on Flash Low voltage supply (hot point)	See VLVD108 in DS[2]
LVD108_P[1] >	6b011001	LVD on PLL Low voltage supply (hot point)	See VLVD108 in DS[2]
LVD108_C[1]	6b011000	LVD on Low voltage internal supply (hot point)	See VLVD108 in DS
HVD140_C[1]	6b010111	LV Core supply high voltage detector (cold point)	See VHVD140 in DS
HVD145_F[1]	6b111001	Flash LV supply high voltage detector (cold point)	See VHVD145 in DS
HVD145_C[1]	6b111000	LV Core supply 2nd high voltage detector (cold point)	See VHVD145 in DS
Internal POR on HV[1]	6b010101	HV PMC supply power-On-Reset detector	~Supply × 1.2 ÷ POR240 _Threshold (falling)[3] (See DS VPOR240)
LVD270_C[1]	6b010100	HV PMC supply low voltage detector	~supply*1.2/LVD270_C (falling)[3] (see VLVD270 in DS)
HVD600_C[1]	6b010011	HV PMC supply high voltage detector	~Supply × 1.2 ÷ HVD600 _Threshold (rising)[3] (See VHVD360 in DS)
LVD270_F[1]	6b010010	HV flash supply low voltage detector	~Supply × 1.2 ÷ LVD270 _F (falling)[3] (see VLVD270 in DS)
LVD295_F[1]	6b010001	HV flash supply low voltage detector	~Supply × 1.2 ÷ LVD295 _C (falling)[3] (see VLVD295 in DS)
HVD360_F[1]	6b010000	HV flash supply high voltage detector	~Supply × 1.2 ÷ HVD360 _Threshold (rising)[3] (See VHVD360 in DS)
LVD295_A[1]	6b001111	HV ADC supply low voltage detector	~Supply × 1.2 ÷ LVD295 _C (falling)[3] (see VLVD295 in DS)
LVD400_A[1]	6b001110	HV ADC supply low voltage detector	~Supply × 1.2 ÷ LVD400 _C (falling)[3] (see VLVD400 in DS)

Table continues on the next page...

Table 7-67. Parameter monitoring via ADC channel (continued)

PMC ADC Mode Signal Description	ADC Channel Select Register	ADC Mode Description	Nominal value expected
HVD600_A[1]	6b001101	HV ADC supply high voltage detector	~Supply × 1.2 ÷ HVD600 _Threshold (rising)[3]
LVD270_IM[1]	6b001100	HV IO main supply low voltage detector	~Supply × 1.2 ÷ LVD270 _Threshold (falling)[3]
LVD360_IM[1]	6b001011	HV IO main supply low voltage detector	~Supply × 1.2 ÷ LVD360 _Threshold (falling)[3]
LVD400_IM[1]	6b001010	HV IO main supply low voltage detector	~Supply × 1.2 ÷ LVD400_Threshold (falling)[3]
Internal POR on flash HV[1]	6b001001	HV flash supply power-On-Reset detector	~Supply × 1.2 ÷ POR250 _Threshold (falling)[3]
LVD270_IF[1]	6b001000	FlexRay I/O supply low voltage detector	~Supply × 1.2 ÷ LVD270 _Threshold (falling)[3]
LVD270_IJ[1]	6b000110	JTAG I/O supply low voltage detector	~Supply × 1.2 ÷ LVD270 _Threshold (falling)[3]
LVD270_O[1]	6b000100	Oscillator supply low voltage detector	~Supply × 1.2 ÷ LVD270 _Threshold (falling)[3]
LVD270_EBI[1]	6b000011	HV IE supply low voltage detector	~Supply × 1.2 ÷ LVD270 _Threshold (falling)[3]
LVD270_IF2[1]	6b000010	Second FlexRay I/O supply low voltage detector	~Supply × 1.2 ÷ LVD270 _Threshold (falling)[3]
Normal Operation Mode[1]	6b000000	ADC channel off	—
—	6b111111	Reserved / ADC channel off	—
VDDREG	6b110111	Scaled VDDREG supply	PMC supply x 1/5
VDD ADC	6b110110	Scaled ADC supply	ADC supply x 1/5
VDD Oscillator	6b110101	Scaled Oscillator supply	Oscillator supply × 17/60
VDD Flash	6b110100	Scaled Flash supply	Flash supply (× 17/60)
VDD main I/O	6b110011	Scaled 5V main I/O supply	5V main I/O supply (x 1/5)
VDD Flexray	6b110010	Scaled Flexray supply	Flexray supply x 17/60
VDD Core	6b110001	Core supply hot Sense	Core supply hot Point
VDD PLL	6b110000	PLL supply Sense	PLL supply Sense
VDD EBI	6b101111	Scaled EBI supply	EBI supply × 17/60
VDD Flexray2	6b101110	Scaled 2nd Flexray supply	2nd Flexray supply × 17/60
VDD LFAST PLL	6b101101	LFAST PLL supply	LFAST PLL supply, 1.30V
VDD IJ	6b101100	Scaled Jtag supply	Jtag supply × 17/60
LV flash supply sense	6b101011	LV flash supply sense point	LV flash supply sense point
Buddy device supply sense	6b101010	Buddy Device supply sense point	Buddy Device supply sense point
LV core supply sense	6b101001	LV core supply cold sense point	LV core supply

Table continues on the next page...

Table 7-67. Parameter monitoring via ADC channel (continued)

PMC ADC Mode Signal Description	ADC Channel Select Register	ADC Mode Description	Nominal value expected
internal voltage	6b100000	—	—
Vref0p45	6b010110	LVD/HVD self test voltage	0.47 V
Vref0p75	6b100111	LVD selftest voltage	0.74 V
Vref1p0	6b100110	LVD selftest voltage	1.0 V
Vref1p5	6b100101	HVD selftest voltage	1.6 V
VDDA22A	6b111100	pre-regulator output	2.5 V
Reserved	6b111011	—	—
Reserved	6b111110	—	—
BG1P20_REF	6b111010	bandgap trimmed reference voltage	1.2 V
Reserved	6b000111	—	—
Vbandgap buffered	6b000101	1.2 V	1.2 V
Reserved	6b100100	—	—
BIS_BG1P20_REF	6b100011	2nd bandgap trimmed reference for flash regulators	1.2 V
Reserved	6b100010	—	—
Reserved	6b100001	—	—
—	6b111101	Internal voltage	Reserved

1. When doing conversions on this channel, it is recommended to mask the corresponding HVD/LVD from generating a reset for the duration of the conversion when this channel is enabled.
2. Refer DS, 'Voltage monitor electrical characteristic Table (footnote 6)' for their values.
3. Bandgap trimmed reference voltage

For additional information see Analog-to-Digital Converters (ADC) Configuration, for an overview of the ADC subsystem, including a block diagram and a summary of analog input pin multiplexing. The chapter also provides configuration details for the SARADCs.

7.12.2 LVD / HVD self test

A self test functionality exists for each LVD / HVD. Programming LVD / HVD self test register to a specific value designated to a LVD /HVD will start the self test for that LVD / HVD. During the self test, the voltage sense input of the LVD /HVD comparator is connected to an internal voltage which is lower than the LVD reference for LVD test (higher than HVD reference for HVD test), so that an artificial assertion of the LVD /HVD comparator can be triggered. After the comparator is triggered, the voltage sense input will be re-connected to the supply it monitors automatically.

>The self test takes less than 5µs from a self test register programmed to start the test to LVD / HVD comparator output changing.

While programming the self test register bits, a timer should be started. If the LVD / HVD status has not been updated when the timer runs out, this indicates the LVD / HVD self test failed. The figure below shows a block diagram of the LVD self test. The table below provides LVD/HVD self test decoding.

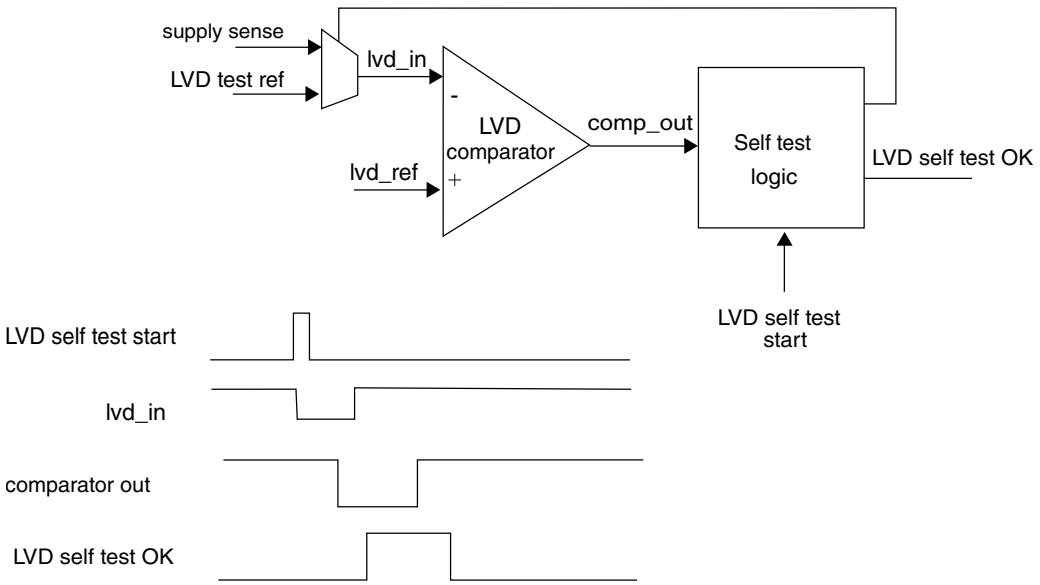


Figure 7-23. LVD self test simplified diagram

Table 7-68. LVD /HVD self test decoding

LVD / HVD under test	LVD self test register
0 - POR085_F	6b011111
1 - POR085_C	6b011110
2 - POR098_C	6b011101
3 - LVD114_C	6b011100
4 - HVD140_C	6b011011
5 - POR098_F	6b011010
6 - LVD114_F	6b011001
7 - (no test)	6b011000
8 - LVD108_B	6b010111
9 - LVD114_P	6b010110
10 - LVD114_H	6b010101
11 - HVD145_C	6b010100
12 -HVD145_F	6b010011
13 -POR081_C	6b010010
14 - LVD280_IF2	6b010001
15 - LVD280_IE	6b010000
16 - LVD280_C	6b001111
17 - POR260_C	6b001110

Table continues on the next page...

Table 7-68. LVD /HVD self test decoding (continued)

LVD / HVD under test	LVD self test register
18 - HVD600_C	6b001101
19 - LVD280_F	6b001100
20 - LVD300_F	6b001011
21 - HVD360_F	6b001010
22 - LVD300_A	6b001001
23 - LVD400_A	6b001000
24 - HVD600_A	6b000111
25 - LVD280_IM	6b000110
26 - LVD360_IM	6b000101
27 - LVD400_IM	6b000100
28 - LVD280_IF	6b000011
29 - LVD280_IJ	6b000010
30 - LVD280_O	6b000001
31 —	Normal mode
32 —	Reserved
33 - LVD250_F	6b111110
34 - POR_INTV1	6b111101
35 - POR_INTV2	6b111100
36 - POR_INTV3	6b111011
37 - POR_INTV4	6b111010
38 - 63 (no test)	6b111001 - 6b100000

7.13 Safety modules

7.13.1 CRC – Number Of Contexts

The number of contexts available for the current CRC computation of multiple data streams of CRC are listed below:

Table 7-69. CRC Contexts

Signal name	Description	crc_0	crc_1
CRC_CNTX_NUM (N)	Number of contexts available for the current CRC computation of multiple data streams	1	1

7.13.2 MEMU configuration

The MEMU is responsible for collection and reporting of error events associated with ECC (Error correction code) logic used on SRAMs and FLASH. When any of the following events occur the MEMU receives an error signal that causes an event to be recorded and corresponding error flags to be set and reported to the FCCU.

7.13.2.1 MEMU system connections

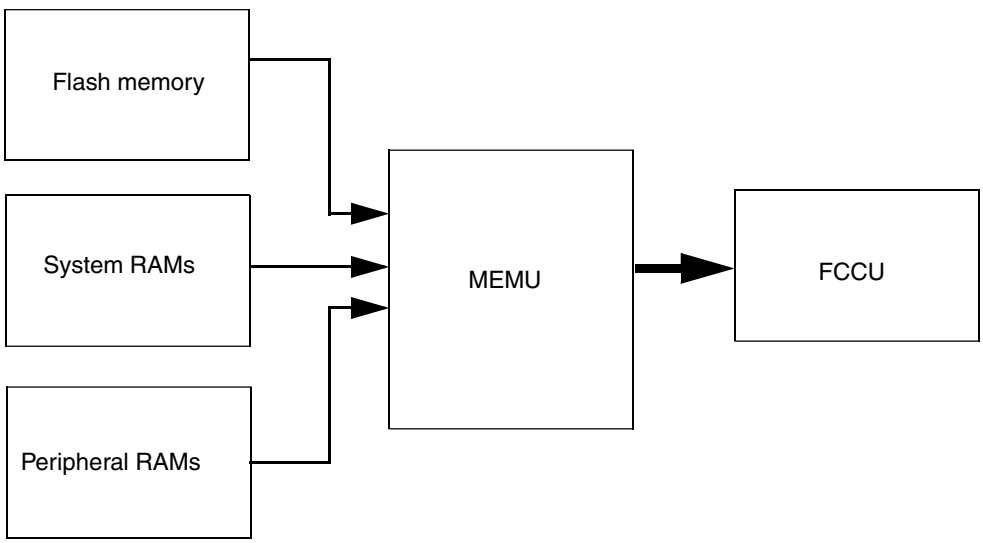


Figure 7-24. MEMU system connections

7.13.2.2 MEMU error sources

The following table shows the MEMU error sources for each instance configured on this chip.

Table 7-70. MEMU error sources

Instance	Value
Number of system RAM unique error sources	102
Number of peripheral RAM unique error sources	33
Number of flash memory unique error sources	3

7.13.2.3 Reporting tables implementation

There are three categories of error reporting for the MEMU:

- Flash memory error reporting
- System RAM error reporting (any RAM that is CPU accessible falls into this category)
- Peripheral RAM error reporting (any RAM that is not CPU accessible falls into this category)

Each category of error reporting has the following tables associated with it:

- Correctable error reporting table
- Uncorrectable error reporting table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 7-71. Number of entries in each reporting table

Error source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash memory	20	1
System RAM	10	1
Peripheral RAM	2	1

When accessing flash memory, the following cases apply:

- Case 1: In the case where a flash memory location has an uncorrectable ECC error, and this location is read by the core, then the error is signaled to the MEMU as two errors:

  a. Flash memory uncorrectable error
  b. RAM uncorrectable error

  In this case, ECC error reporting is performed by the masters, not the slaves. Therefore, in the case of the core reading from memory with the uncorrectable ECC error, the error is reported as a SYSTEM_RAM_UNCERR uncorrectable error. In addition to the core reporting a SYSTEM_RAM error, the flash memory module does its own reporting, indicated by a MEMU FLASH_UNCERR uncorrectable error.

- Case 2: In the case where a flash memory location is read by the eDMA, and an uncorrectable ECC error is encountered, then this error is signaled to the MEMU also as two errors:

  a. Flash memory uncorrectable error
  b. Peripheral uncorrectable error

  In this case, the eDMA is the master, and it reports its errors in the PERIPHERAL_RAM MEMU error reporting tables. Since flash memory is being accessed, the flash memory module also reports a MEMU error in the FLASH_UNCERR table.

- Case 3: In the case where a correctable error in the flash memory is read by a CPU core or the eDMA module, the following occurs:

  - No MEMU reporting by the core or the eDMA
  - Correctable flash memory error signaled to the MEMU

  When the flash memory controller detects a single-bit inversion, it signals the event to the MEMU and corrects the data before providing it back to the system (core or eDMA). So, as far as the requesting master is concerned, it should only “see” correct data, with no indication on their part that a single-bit correction occurred.

7.13.2.4 Concurrent overflow register (OFLW) implementation

Errors are reported to the MEMU from these categories:

- Flash memory
- System RAM
- Peripheral RAM

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Positions not explicitly listed in the tables below are reserved.

Table 7-72. Corresponding error sources for system RAM OFLW0 register

OFLW0 bit-field position (bit)	Error sources
0	z4a Data AHB
1	z4a Instruction AHB
2	z4a Data TCM (DMEM)
3	z4a Instruction TCM (IMEM)

Table continues on the next page...

Table 7-72. Corresponding error sources for system RAM OFLW0 register (continued)

OFLW0 bit-field position (bit)	Error sources
4	z4a Data Cache
5	z4a Instruction Cache
6	z4b Data AHB
7	z4b Instruction AHB
8	z4b Data TCM (DMEM)
9	z4b Instruction TCM (IMEM)
10	z4b Data Cache
11	z4b Instruction Cache
12	z4c Data AHB
13	z4c Instruction AHB
14	z4c Data TCM (DMEM)
15	z4c Instruction TCM (IMEM)
16	z4c Instruction Cache
17	SRAM
18	Decorated PBRIDGE0
19	Decorated PBRIDGE1
20	Decorated PRAM
21	Decorated z4a TCM
22	Decorated z4b TCM
23	Decorated z4c TCM
24	MBIST z4a Instruction TCM
25	MBIST z4a Data TCM Instance 1
26	MBIST z4a Data TCM Instance 2
27	MBIST z4a Instruction Cache Instance 1
28	MBIST z4a Instruction Cache Instance 2
29	MBIST z4a Data Cache Instance 1
30	MBIST z4a Data Cache Instance 2
31	MBIST z4a Instruction Cache Tag

Table 7-73. Corresponding error sources for system RAM OFLW1 register

OFLW1 bit-field position (bit)	Error Sources
0	MBIST z4a Data Cache Tag
1	MBIST z4b Instruction TCM
2	MBIST z4b Data TCM Instance 1
3	MBIST z4b Data TCM Instance 2
4	MBIST z4b Instruction Cache Instance 1
5	MBIST z4b Instruction Cache Instance 2
6	MBIST z4b Data Cache Instance 1

Table continues on the next page...

Table 7-73. Corresponding error sources for system RAM OFLW1 register (continued)

OFLW1 bit-field position (bit)	Error Sources
7	MBIST z4b Data Cache Instance 2
8	MBIST z4b Instruction Cache Tag
9	MBIST z4b Data Cache Tag
10	MBIST z4c Instruction TCM Instance 1
11	MBIST z4c Instruction TCM Instance 2
12	MBIST z4c Data TCM Instance 1
13	MBIST z4c Data TCM Instance 2
14	MBIST z4c Instruction Cache Instance 1
15	MBIST z4c Instruction Cache Instance 2
16	MBIST z4c Instruction Cache Instance 3
17	MBIST z4c Instruction Cache Instance 4
18	MBIST z4c Instruction Cache Tag
19	MBIST RAM 0
20	MBIST RAM 1 Instance 1
21	MBIST RAM 1 Instance 2
22	MBIST Instruction Cache Instance 1
23	MBIST Instruction Cache Instance 2
24	MBIST Instruction Cache Tag Instance 1
25	MBIST Instruction Cache Tag Instance 2
26	MBIST HSM Crypto Channel Controller (C3)
27	MBIST DMA instance 1
28	MBIST DMA instance 2
29	MBIST FlexRay Data RAM instance 1
30	MBIST FlexRay Data RAM instance 2
31	MBIST FlexRay0 LUT RAM instance 1

Table 7-74. Corresponding error sources for system RAM OFLW2 register

OFLW2 bit-field position (bit)	Error Sources
0	MBIST FlexRay0 LUT RAM instance 2
1	MBIST FlexRay1 LUT RAM instance 1
2	MBIST FlexRay1 LUT RAM instance 2
3	MBIST Controller Area Network
4	MBIST GTM FIFO 0
5	MBIST GTM FIFO 1
6	MBIST GTM RAM0 Instance1
7	MBIST GTM RAM0 Instance 2
8	MBIST GTM RAM0 Instance 3
9	MBIST GTM RAM0 Instance 4

Table continues on the next page...

Table 7-74. Corresponding error sources for system RAM OFLW2 register (continued)

OFLW2 bit-field position (bit)	Error Sources
10	MBIST GTM RAM0 Instance 5
11	MBIST GTM RAM0 Instance 6
12	MBIST GTM RAM1 Instance 1
13	MBIST GTM RAM1 Instance 2
14	MBIST GTM RAM1 Instance 3
15	MBIST GTM RAM1 Instance 4
16	MBIST GTM RAM1 Instance 5
17	MBIST GTM RAM1 Instance 6
18	MBIST Digital PLL Instance 1a
19	MBIST Digital PLL Instance 1b
20	MBIST Digital PLL Instance 2
21	MBIST SRAM Instance 1
22	MBIST SRAM Instance 2
23	MBIST SRAM Instance 3
24	MBIST SRAM Instance 4
25	MBIST SRAM Instance 5
26	MBIST SRAM Instance 6
27	MBIST SRAM Instance 7
28	MBIST Nexus Instance 1
29	MBIST Nexus Instance 2
30	MBIST Nexus Instance 3
31	MBIST Nexus Instance 4

Table 7-75. Corresponding error sources for system RAM OFLW3 register

OFLW3 bit-field position (bit)	Error Sources
0	Reserved
1	MBIST FEC FIFO
2	MBIST FEC MIB
3	MBIST Overlay Instance 0
4	MBIST Overlay Instance 1
5	EBI RAM

Table 7-76. Corresponding error sources for peripheral RAM OFLW0 register

OFLW0 bit-field position (bit)	Error Sources
0	DMA TCD 0
1	DMA TCD 1
2	Advanced Memory Unit RAM

Table continues on the next page...

Table 7-76. Corresponding error sources for peripheral RAM OFLW0 register (continued)

OFLW0 bit-field position (bit)	Error Sources
3	Concentrator 0
4	Concentrator 1
5	PBRIDGE0
6	PBRIDGE1
7	FEC RIF RAM
8	FEC MIB RAM
9	HSM Platform RAM
10	HSM Instruction Cache RAM
11	FlexRay0 Data RAM
12	FlexRay0 LUT RAM
13	FlexRay1 Data RAM
14	FlexRay1 LUT RAM
15	CAN RAM
16	GTM FIFO_0
17	GTM FIFO_1
18	GTM Digital PLL r1a
19	GTM Digital PLL r1b
20	GTM Digital PLL r2
21	GTM Multi Channel Sequencer0 core0
22	GTM Multi Channel Sequencer1 core0
23	GTM Multi Channel Sequencer0 core1
24	GTM Multi Channel Sequencer1 core1
25	GTM Multi Channel Sequencer0 core2
26	GTM Multi Channel Sequencer1 core2
27	GTM Multi Channel Sequencer0 core3
28	GTM Multi Channel Sequencer1 core3
29	GTM Multi Channel Sequencer0 core4
30	GTM Multi Channel Sequencer1 core4
31	GTM Multi Channel Sequencer0 core5

Table 7-77. Corresponding error sources for peripheral RAM OFLW1 register

OFLW1 bit-field position (bit)	Error Sources
0	GTM Multi Channel Sequencer1 core5

Table 7-78. Corresponding error sources for flash memory OFLW register

OFLW0 bit-field position (bit)	Error sources
0	Flash 0

Table continues on the next page...

Table 7-78. Corresponding error sources for flash memory OFLW register (continued)

OFLW0 bit-field position (bit)	Error sources
1	Flash 1
2	Reserved

7.13.2.5 FlexRay lookup table (LUT) RAM alignment

Table 7-79. FlexRay LUT RAM alignment

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register name	MEMU error reporting (PERIPH_RAM_*_ADDR)
0	FFE5_0800	FFE5_080C	FR_MBCCFR0 FR_MBFIDR0 FR_MBIDXR0 FR_MBCCFR1 FR_MBFIDR1 FR_MBIDXR1	FFE5_0802
1	FFE5_0810	FFE5_081C	FR_MBCCFR2 FR_MBFIDR2 FR_MBIDXR2 FR_MBCCFR3 FR_MBFIDR3 FR_MBIDXR3	FFE5_0812
2	FFE5_0820	FFE5_082C	FR_MBCCFR4 FR_MBFIDR4 FR_MBIDXR4 FR_MBCCFR5 FR_MBFIDR5 FR_MBIDXR5	FFE5_0822
3	FFE5_0830	FFE5_083C	FR_MBCCFR6 FR_MBFIDR6 FR_MBIDXR6 FR_MBCCFR7 FR_MBFIDR7 FR_MBIDXR7	FFE5_0832
4	FFE5_0840	FFE5_084C	FR_MBCCFR8 FR_MBFIDR8 FR_MBIDXR8 FR_MBCCFR9 FR_MBFIDR9 FR_MBIDXR9	FFE5_0842
5	FFE5_0850	FFE5_085C	FR_MBCCFR10 FR_MBFIDR10 FR_MBIDXR10 FR_MBCCFR11 FR_MBFIDR11 FR_MBIDXR11	FFE5_0852
6	FFE5_0860	FFE5_086C	FR_MBCCFR12 FR_MBFIDR12 FR_MBIDXR12 FR_MBCCFR13 FR_MBFIDR13 FR_MBIDXR13	FFE5_0862
7	FFE5_0870	FFE5_087C	FR_MBCCFR14 FR_MBFIDR14 FR_MBIDXR14 FR_MBCCFR15 FR_MBFIDR15 FR_MBIDXR15	FFE5_0872

Table continues on the next page...

Table 7-79. FlexRay LUT RAM alignment (continued)

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register name	MEMU error reporting (PERIPH_RAM_*_ADDR)
8	FFE5_0880	FFE5_088C	FR_MBCCFR16 FR_MBFIDR16 FR_MBIDXR16 FR_MBCCFR17 FR_MBFIDR17 FR_MBIDXR17	FFE5_0882
9	FFE5_0890	FFE5_089C	FR_MBCCFR18 FR_MBFIDR18 FR_MBIDXR18 FR_MBCCFR19 FR_MBFIDR19 FR_MBIDXR19	FFE5_0892
...	...	...	...	...
62	FFE5_0BE0	FFE5_0BEC	FR_MBCCFR125 FR_MBFIDR125 FR_MBIDXR125 FR_MBCCFR124 FR_MBFIDR124 FR_MBIDXR124	FFE5_0BE2
63	FFE5_0BF0	FFE5_0BFC	FR_MBCCFR127 FR_MBFIDR127 FR_MBIDXR127 FR_MBCCFR126 FR_MBFIDR126 FR_MBIDXR126	FFE5_0BF2

7.13.2.6 FlexRay lookup table (LUT) RAM misalignment

Table 7-80. FlexRay LUT RAM misalignment

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register names	MEMU error reporting (PERIPH_RAM_*_ADDR)
64	FFE5_1000	FFE5_100A	MBDOR0..5	FFE5_1000
65	FFE5_100C	FFE5_1016	MBDOR6..11	FFE5_1010
66	FFE5_1018	FFE5_1022	MBDOR12..17	FFE5_1020
67	FFE5_1024	FFE5_102E	MBDOR18..23	FFE5_1030
68	FFE5_1030	FFE5_103A	MBDOR24..29	FFE5_1040
69	FFE5_103C	FFE5_1046	MBDOR30..35	FFE5_1050
70	FFE5_1048	FFE5_1052	MBDOR36..41	FFE5_1060
71	FFE5_1054	FFE5_105E	MBDOR42..47	FFE5_1070
72	FFE5_1060	FFE5_106A	MBDOR48..53	FFE5_1080
73	FFE5_106C	FFE5_1076	MBDOR54..59	FFE5_1090
74	FFE5_1078	FFE5_1082	MBDOR60..65	FFE5_10A0

Table continues on the next page...

Table 7-80. FlexRay LUT RAM misalignment (continued)

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register names	MEMU error reporting (PERIPH_RAM_*_ADDR)
75	FFE5_1084	FFE5_108E	MBDOR66..71	FFE5_10B0
76	FFE5_1090	FFE5_109A	MBDOR72..77	FFE5_10C0
77	FFE5_109C	FFE5_10A6	MBDOR78..83	FFE5_10D0
78	FFE5_10A8	FFE5_10B2	MBDOR84..89	FFE5_10E0
79	FFE5_10B4	FFE5_10BE	MBDOR90..95	FFE5_10F0
80	FFE5_10C0	FFE5_10CA	MBDOR96..101	FFE5_1100
81	FFE5_10CC	FFE5_10D6	MBDOR102..107	FFE5_1110
82	FFE5_10D8	FFE5_10E2	MBDOR108..113	FFE5_1120
83	FFE5_10E4	FFE5_10EE	MBDOR114..119	FFE5_1130
84	FFE5_10F0	FFE5_10FA	MBDOR120..125	FFE5_1140
85	FFE5_10FC	FFE5_1106	MBD0R126..131	FFE5_1150
86	FFE5_1108	FFE5_1112	LEETR0..5	FFE5_1160

7.13.3 Indirect Memory Access (IMA) configuration

The IMA refers to the alteration of memory data values during system development and test activities. This section provides details about what memory locations can be accessed —either via processor core access or via the IMA module—and how they are configured.

Note

See the *Microcontroller Security Reference Manual* for the HSM IMA details.

7.13.3.1 SRAM Arrays accessible by processor cores

All the SRAM arrays in Table 7-81 can be accessed by the main CPUs (CPU0 & CPU1) and the IOP (CPU2).

Table 7-81. SRAM with processor Core access

Block	Function	Words	Bits	ECC Bits
Safety Core	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	1024	72	—
		1024	72	—

Table continues on the next page...

Table 7-81. SRAM with processor Core access (continued)

Block	Function	Words	Bits	ECC Bits
		—	—	—
		—	—	—
	Dcache	256	80	—
		256	80	—
		—	—	—
		—	—	—
	ITAG	256	64	—
	DTAG	64	72	—
Comp Core	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	1024	72	—
		1024	72	—
		-	-	—
		-	-	—
	Dcache	256	80	—
		256	80	—
		-	-	—
		-	-	—
	ITAG	256	64	—
	DTAG	64	72	—
IOP Core	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	ITAG	128	65	—
System RAM	System RAM	8192	72	—
		8192	72	—
		8192	72	—
		8192	72	—
		8192	72	—
		8192	72	—
Overlay	Overlay	1024	72	—
		1024	72	—

7.13.3.2 Memory accessible via the IMA module

The SRAM arrays in Table 7-82 are accessible via the IMA Module.

Table 7-82. SRAM arrays accessible via IMA

IMA ARRAY SELECT	Block	Function	Words	Bits	ECC Bits
0	None	IMA Disabled	—	—	—
1	DMA	DMA	256	72	71–64
2			256	72	71–64
3	FlexRAY	DRAM	128	26	25–16
4			128	26	25–16
5		LRAM	87	63	in lram_1
6			87	63	63–34
7			87	63	in lram_1
8			87	63	63–34
9	TTCAN	TTCAN	5120	39	38–32
10	GTM	FIFO	1024	36	35–29
11			1024	36	35–29
12		MCS RAM0	1024	39	38–32
13			1024	39	38–32
14			1024	39	38–32
15			1024	39	38–32
16			1024	39	38–32
17			1024	39	38–32
18		MCS RAM1	512	39	38–33
19			512	39	38–33
20			512	39	38–33
21			512	39	38–33
22			512	39	38–33
23			512	39	38–33
24		DPLL RAM1a	128	31	30–24
25		DPLL RAM1b	384	31	30–24
26		DPLL RAM2	4096	31	30–24
27	Reserved	—	—	—	—
28	FEC	FIFO (RIF)	128	44	43–36
29		MIB	64	40	39–32

7.13.3.3 Memory with inaccessible ECC bits

The following table shows memory locations that cannot be accessed.

Table 7-83. No ECC bit access

IMA ARRAY SELECT	Block	Function	Words	Bits	ECC Bits
No IMA Access Required–Debug function					
NAR	NAR	64	128	—	
		64	128	—	
		64	128	—	
		64	128	—	

7.13.4 FCCU configuration

Note

See FCCU failure inputs for a list FCCU failure inputs.

7.13.4.1 FCCU register availability and reset values

Table 7-84 shows the availability and reset values of selected FCCU registers.

Table 7-84. Availability and reset values of selected FCCU registers

Register	Reset value
FCCU_MCS	0x0000_8083
FCCU_RF_CFG0	0xFFFF_FFFF
FCCU_RF_CFG1	0x00FF_FFFF
FCCU_RF_CFG2	0x0000_0000
FCCU_RF_CFG3	0x0000_0000
FCCU_RFS_CFG0	0x0200_2000
FCCU_RFS_CFG1	0x0000_0000
FCCU_RFS_CFG2	0x0000_0000
FCCU_RFS_CFG3	0x0000_0000
FCCU_RFS_CFG4	0x0000_0000
FCCU_RFS_CFG5	0x0000_0000
FCCU_RFS_CFG6	0x0000_0000
FCCU_RFS_CFG7	0x0000_0000
FCCU_RF_E0	0x0000_1040
FCCU_RF_E1	0x0000_0000
FCCU_RF_E2	0x0000_0000
FCCU_RF_E3	0x0000_0000
FCCU_RF_TOE0	0x0000_0000

Table continues on the next page...

Table 7-84. Availability and reset values of selected FCCU registers (continued)

Register	Reset value
FCCU_RF_TOE1	0x0000_0000
FCCU_RF_TOE2	0x0000_0000
FCCU_RF_TOE3	0x0000_0000

7.13.4.2 FOSU parameters

The FOSU_COUNT, or FOSU time out response time, is 8000 safe clock cycles.

7.13.4.3 Available FCCU output signals

Table 7-85 shows the available FCCU output signals implemented on this chip.

For more details refer to FCCU failure inputs.

Table 7-85. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0]	F0
EOUT[1]	F1

7.13.4.4 Error signal flow diagram

In the following diagram, the FCCU_RF registers are labeled as FCCU_NCF registers.

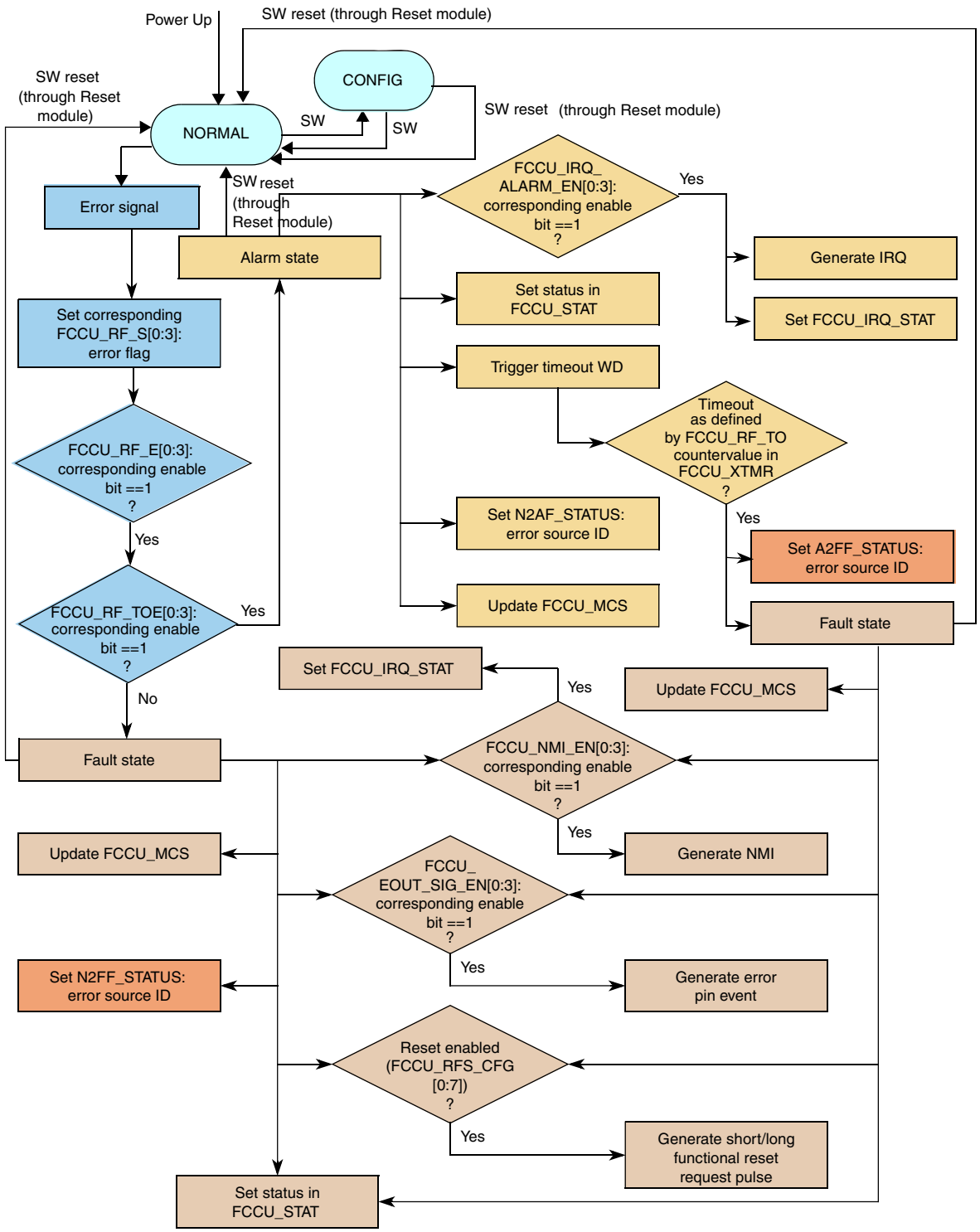


Figure 7-25. Error signal flow

7.13.4.5 FCCU failure inputs

Table 7-86 shows the different failure input signals of the FCCU, how they can be tested, and their default configuration after reset[1].

1. All of these faults require a long functional reset.

Table 7-86. FCCU failure inputs[1]

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
0	TEMP_ERROR	Event indication from temperature sensor. The fault is cleared by clearing the status in PMC_dig_EPR_TD, followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	Not testable	No reaction	PMC_dig_EPR_TD Event Pending Register (PMCDIG_EPR_TD)	Correction of violating condition and power reset cycle
1	LVD_ERROR	Voltage out of range indication from LVDs. The fault is cleared by clearing the status in PMC_dig_EPR_VD*n*, followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LVD assertion test by tap point manipulation	No reaction	PMC_dig_EPR_VD3 PMC_dig_EPR_VD4 PMC_dig_EPR_VD9 PMC_dig_EPR_VD10 PMC_dig_EPR_VD13 PMC_dig_EPR_VD14 PMCDIG_memoryMap	Correction of violating condition and power reset cycle
2	HVD_ERROR	Voltage out of range indication from HVDs. The fault is cleared by clearing the status in PMC_dig_EPR_VD*n*, followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	HVD assertion test by tap point manipulation	No reaction	PMC_dig_EPR_VD7 PMC_dig_EPR_VD8 PMC_dig_EPR_VD12 PMC_dig_EPR_VD15 PMCDIG_memoryMap	Correction of violating condition and power reset cycle
3	DPMC_DCF_SAFETY_ERR	Digital PMC initialization error during DCF data load (triple-vote mismatch). The fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]. The status gets cleared if the fault is not persistent.	LBIST	No reaction	Power Management Controller digital interface (PMC_dig)	POR, Destructive Reset
4	MEMORY_DCF_SAFETY_ERR	Indication of fault during timing/ repair configuration initialization by SSCM of different memories. The status is cleared by clearing SSCM_STATUS[CER], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]. The	LBIST	No reaction	SSCM_memoryMap	POR, Destructive Reset

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
		status gets cleared if the fault is not persistent. Additionally, this error will cause an MBIST to fail.				
5	SSCM_XFER_FLASH_ERR	SSCM transfer error (during the SSCM and STCU DCF loading), or flash memory initialization error.	Testable (and wrong DCF configuration or spurious data from Flash can trigger other failures)	No reaction	SSCM_memoryMap	POR, Destructive Reset
6	STCU_UF	STCU2 unrecoverable fault indication during self-test. It is simultaneously reported to the MC_RGM which will initiate a destructive reset in case of an offline self-test. In the STCU2, self-test failures can be configured as recoverable or unrecoverable faults which would then lead to assertion of the corresponding fault line to the FCCU. Fault is cleared by clearing STCU2_ERR_STAT[UFSF] followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	SW check of BIST MISR or Feedback loop via FCCU Fake bit	Destructive reset by MC_RGM directly, during offline self-test	ERR_STAT Reset sources 'Destructive' Event Reset Disable Register (MC_RGM_DERD)	POR, Destructive reset. During offline test, assertion of this fault will automatically cause a destructive reset request by MC_RGM.
7	STCU_RF	STCU2 recoverable fault indication during self-test. In the STCU2, self-test failures can be configured as recoverable or unrecoverable faults which would then lead to assertion of the corresponding fault line to the FCCU. Fault is cleared by clearing	SW check of BIST MISR or Feedback loop via FCCU Fake bit	No reaction	ERR_STAT	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
		STCU2_ERR_STAT[RFSF] followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].				
8	STCU_LMBIST_USR_ERR [4]	Activation of LBIST or MBIST control during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]. The status will clear if the fault is not persistent.	Feedback loop via FCCU Fake bit.	No reaction	—	Long Functional Reset to reset STCU, POR, Destructive Reset
9	SPURIOUS_DEBUG_SSCM_ACTIVATION	JTAG, NPC or Debug functionality moved out of reset when JCOMP is low or activation of SSCM or DCF client during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Feedback loop via FCCU Fake bit	No reaction	—	Long Functional Reset
10	RCCU_0	Safety cores out of sync during lockstep (RCCU0a). Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Feedback loop via FCCU Fake bit connected to additional RCCU input	No reaction	Redundancy Control and Checker Units (RCCUs)	Functional Reset
11	RCCU_1	Safety cores out of sync during lockstep ((RCCU0b ... RCCU0e, RCCU1)). Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Feedback loop via FCCU Fake bit connected to additional RCCU input	No reaction	Redundancy Control and Checker Units (RCCUs)	Functional Reset

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
12	SWT2_ResetReq	Reset request from Software Watchdog of Peripheral Core_2 domain. The fault is cleared by clearing FCCU_RF_S*n*[RFS*m*] after the status has been cleared inside SWT2.	LBIST + SW test by not triggering SWT during startup	Long Reset	Software Watchdog Timer (SWT)	Functional Reset (application dependent)
13	SWT1_ResetReq	Reset request from Software Watchdog of Main Core_1 domain. The fault is cleared by clearing FCCU_RF_S*n*[RFS*m*] after the status has been cleared inside SWT1.	LBIST + SW test by not triggering SWT during startup	No reaction	Software Watchdog Timer (SWT)	Functional Reset (application dependent)
14	SWT0_InterrutpReq	First timeout interrupt request from Software Watchdog of Safety Core (Main Core_0). Fault is cleared by clearing SWT0_SWT_IR[TIF], followed by clearing the FCCU channel status FCCU_RF_S*n*[RFS*m*].	LBIST + SW test by not triggering SWT during startup	No reaction	Software Watchdog Timer (SWT)	Functional Reset (application dependent)
15	SWT0_ResetReq	Second timeout (reset) request from Software Watchdog of Safety Core (Main Core_0). The fault is cleared by clearing FCCU_RF_S*n*[RFS*m*] after the status has been cleared inside SWT0.	LBIST + SW test by not triggering SWT during startup	No reaction	Software Watchdog Timer (SWT)	Functional Reset
16	MEMU_RAM_CE	Indication by MEMU of occurrence of correctable errors in system RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[SR_CE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Injection of RAM ECC error via ECC bit access and later reading	No reaction	Error flag register (MEMU_ERR_FLAG)	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
17	MEMU_RAM_UCE	Indication by MEMU of occurrence of uncorrectable errors in system RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_UCE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Injection of RAM ECC error via ECC bit access and later reading	No reaction	Error flag register (MEMU_ERR_FLAG)	—
18	MEMU_RAM_BOV[5, 6]	Indication of overflow of system RAM ECC buffer or table in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_EBO, SR_CEO or SR_UCO], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Functional test[7]	No reaction	Error flag register (MEMU_ERR_FLAG)	—
19	MEMU_PER_CE[5, 6]	Indication by MEMU of the occurrence of correctable errors in peripheral RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[PR_CE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]	LBIST + Functional test [8]	No reaction	Error flag register (MEMU_ERR_FLAG)	—
20	MEMU_PER_UCE[5, 6]	Indication by MEMU of the occurrence of uncorrectable errors in peripheral RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[PR_UCE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]	LBIST + Functional test [9]	No reaction	Error flag register (MEMU_ERR_FLAG)	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
21	MEMU_PER_BOV[5, 6]	Indication of overflow of peripheral RAM ECC buffer or table in MEMU. The fault is cleared by clearing MEMU_ERR_FLAG[PR_EBO, PR_CEO or PR_UCO], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]	LBIST + Functional test [10]	No reaction	Error flag register (MEMU_ERR_FLAG)	—
22	MEMU_FLS_CE[6]	Indication by MEMU of occurrence of correctable errors in flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_CE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Reading preprogrammed correctable ECC error from UTEST flash memory	No reaction	Error flag register (MEMU_ERR_FLAG)	—
23	MEMU_FLS_UCE[6]	Indication by MEMU of occurrence of uncorrectable errors in flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_UCE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Reading preprogrammed uncorrectable ECC error from UTEST flash memory	No reaction	Error flag register (MEMU_ERR_FLAG)	—
24	MEMU_FLS_BOV[5, 6]	Indication of overflow of flash memory ECC buffer or table in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[F_EBO, F_CEO or F_UCO], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Functional test[11]	No reaction	Error flag register (MEMU_ERR_FLAG)	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
25	IMA_SOC	MCU IMA active . Fault is cleared by selecting zero for IMA_SLCT[ARRAY_SLCT], followed by clearing FCCU_RF_Sn[RFSm].	LBIST + IMA read	No reaction	—	—
26	SMPU_MONITOR	Indication of fault in SMPU_0 or SMPU_1 logic fault resulting in an incorrectly refused access, but failure to indicate the error to the requesting bus master. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	LBIST	No reaction	—	—
27	Reserved	—	—	—	—	—
28	Reserved	—	—	—	—	—
29	FM_PLL_0	PLL0 loss of lock. In order for this fault source to be asserted in the event of loss of lock, the Loss of Lock needs to be enabled by writing (PLLDIG_PLL0CR[LOLIE]) = 1. Fault is cleared by clearing PLLDIG_PLL0SR[LOLF], followed by clearing FCCU_RF_S*n*[RFS*m*].	LBIST + Forced loss by PLL reconfiguration	No reaction	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)	application integrity checks
30	FM_PLL_1	PLL1 loss of lock. In order for this fault source to be asserted, the Loss of Lock indication needs to be enabled by writing PLLDIG_PLL1CR[LOLIE] = 1. Fault is cleared by clearing PLLDIG_PLL1SR[LOLF], followed by clearing FCCU_RF_S*n*[RFS*m*].	LBIST + Forced loss by PLL reconfiguration	No reaction	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)	application integrity checks
31	CMU_0_OSC [-1, 12]	Loss of XOSC clock. The fault is cleared by clearing CMU_ISR[OLRI] of CMU_0,	LBIST	No reaction	CMU Interrupt Status Register (CMU_ISR)	Functional reset

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
		followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].				
32	CMU_0_PLL [15][12]	SYSCLK frequency out of range. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]	LBIST + Test by misconfiguration [15]	No reaction	CMU Interrupt Status Register (CMU_ISR)	Functional reset
33	CMU_comp_subsys[12], [16]	Exception from monitoring internal clocks of comp_subsys. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_*n*, followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]	LBIST + Test by misconfiguration [15]	No reaction	CMU Interrupt Status Register (CMU_ISR)	Functional reset
34	CMU_other [12, 17]	Exception from monitoring internal clocks of other subsystems. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_*n*, followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*]	LBIST + Test by misconfiguration [15]	No reaction	CMU Interrupt Status Register (CMU_ISR)	Functional reset
35	PERIPH_XBIC_GASKET_MONITOR	Indication of a hardware fault (detected by the EDC_after_ECC) resulting in corrupted transaction through the peripheral XBAR or gaskets handling communication from Safety core to PERIPH_XBAR and back. Fault is	LBIST	No reaction	Crossbar Integrity Checker (XBIC)	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
		cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].				
36	EDC_ECC_FLASH	Indication of a hardware fault (detected by the EDC_after_ECC) in the ECC logic of the flash memory resulting in corrupted ECC detection/correction. Fault is cleared by a dummy read of the flash memory followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	Module Configuration Register (C55FMC_MCR)	—
37	EDC_ ECC_FLASH_C	Indication of hardware fault in the flash memory controller resulting in corrupted ECC detection / correction. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	e2eECC and flash accesses	—
38	ENC_ERR_FLASH	Indication of hardware fault resulting in corrupted flash memory access, or Read Voltage or Read Reference were out of range for a previous read. Fault is cleared by clearing MCR[RVE] and MCR[RRE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	—	—
39	ADDR_FDBK_ERR_FLASH_C	Indication of an addressing error in the flash memory controller resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	—	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
40	COMP_XBIC_DSMC_Monitor	Indication of an addressing or control fault resulting in corrupted transaction through the computational XBAR or interference by system RAM DSMC. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	Crossbar Integrity Checker (XBIC)	—
41	ADDR_FDBK_ERR_CAL_RAM[4]	Indication of an addressing error in the on-chip overlay RAM resulting in corrupted access to on-chip overlay RAM. Status should be cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction	Flash address generation check	—
42	SAFE_CAL_ERR	Indication of hardware fault in the flash memory controller resulting in corrupted overlay remap evaluation. Status should be cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	Flash address generation check	—
43	ADDR_FDBK_ERR_PRAM + RAM_LWB_ERR	Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	Transaction monitor	—
44	SMPU_ERROR	Indication of the prevention of an unauthorized access by SMPU_0 or SMPU_1. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + willfully causing an access to an unauthorized location	No reaction	System Memory Protection Unit (SMPU)	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
45	EDC_ERR_PRAM	Indication of a fault in the controller PRAM of the system RAM resulting in a potentially corrupted RAM word during RMW access or stale word during read. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	LBIST	No reaction		—
46	DFT1	Test circuitry Group 1 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	—	—
47	DFT2	Test circuitry Group 2 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	—	—
48	DFT3	Test circuitry Group 3 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	—	—
49	DFT4	Test circuitry Group 4 activation. Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST	No reaction	—	—
50	Safety Core Machine Check Exception[18]	Safety Core exception indication. The fault is cleared by clearing the status in MCSR register of the Safety core followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Willfully cause an appropriate exception (for example, access a non-existing address)	No reaction		—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
51	Lockstep mode	Indication of disabled Checker Core or RCCUs.Fault is cleared by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + Feedback loop via FCCU Fake bit	No reaction	Dual-core lockstep	—
52	Safe_Mode	Indication of safe mode request assertion from MC_RGM. The event is also captured in MC_ME and available as status for crosscheck. Status should be cleared first in the MC_RGM_FES, then followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	LBIST + (in principle requesting an RGM "safe mode" though that is not typical, and should be avoided)	No reaction	'Functional' Event Status Register (MC_RGM_FES)	—
53	Compensation Disable	Indication of undesired deactivation of pad compensation. Fault may effect a distorted transmissions on I/Os associated with VDD_HV_IO_EBI, VDD_HV_IO_FLEX, and VDD_HV_IO_FLEXE. The indication is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	—	No reaction	—	—
54	EIN_ERR[19, 20]	Error input pin (FI[0], external of the MCU) signalling an error condition. The fault is cleared by clearing the fault driving FI[0] followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].	External activation of EIN. External entity has to pull the error input low.	No reaction	—	—

Table continues on the next page...

Table 7-86. FCCU failure inputs[1] (continued)

FCCU channel [2]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery Mechanism[3]
55	SWT3_ResetReq	Reset request from Security Watchdog. The fault is cleared by clearing FCCU_RF_S*n*[RFS*m*] after the status has been cleared in the SWT3.	LBIST+ SW test by not triggering SWT during startup	No Reaction		—

1. All of these faults require a long functional reset.
2. See "Fault Collection and Control Unit (FCCU)" chapter for channel mapping details.
3. During execution of the safety function. Application level reactions not limited to FCCU reactions
4. Failure will be asserted during on-line MBIST.
5. Error reaction path check contents subject to change.
6. Fake fault cannot be generated for FCCU channel (these faults are injected only through the MEMU).
7. Assert the fault line by writing 1, followed by 0, to MEMU_DEBUG[FR_SR_EBO]. Deassert by writing 1 to MEMU_ERR_FLAG[SR_EBO], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].
8. Assert the fault line by writing 1, followed by 0, to MEMU_DEBUG[FR_PR_CE]. Deassert by writing 1 to MEMU_ERR_FLAG[PR_CE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].
9. Assert the fault line by writing 1, followed by 0, to MEMU_DEBUG[FR_PR_UCE]. Deassert by writing 1 to MEMU_ERR_FLAG[PR_UCE], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].
10. Assert the fault line by writing 1, followed by 0, to MEMU_DEBUG[FR_PR_EBO]. Deassert by writing 1 to MEMU_ERR_FLAG[PR_EBO], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].
11. Assert the fault line by writing 1, followed by 0, to MEMU_DEBUG[FR_F_EBO]. Deassert by writing 1 to MEMU_ERR_FLAG[F_EBO], followed by clearing the FCCU channel status, FCCU_RF_S*n*[RFS*m*].
12. See Clock generation figure (Figure 1) in the Clocking chapter ().
13. CMU_0 OLR event.
14. CMU_0 OLR event.
15. CMU_0 FLL and FHH events.
16. CMU_1, CMU_2, CMU_3, CMU_11, FLL, and FHH events.
17. CMU_4, CMU_5, CMU_6, CMU_7, CMU_8, CMU_9, CMU_10, FLL and FHH events.
18. Alarm #50 is asserted by Main Core_0 when a machine check condition has caused an “Async Mchk” or “Error Report” type syndrome bit to be set in the Machine Check Syndrome register (Refer to Section 14.7.4, “Machine Check Syndrome Register (MCSR)”). See "Exceptions and Conditions" table in the *e200z4 Core Reference Manual* for all exceptions which can occur in the core and table "Error report machine check exceptions" for errors indicated by a machine check exception. Note that the following do not cause a machine check and thus do not signal to the FCCU: access blocked by MPU, wrongly aligned access, execution of a privileged or trap instruction, FPU exceptions. Such errors will have to be handled by the respective exception handler if they are safety-relevant.
19. EIN_ERR can be used only when bistable protocol is selected for external error indication, and the FI[0] pin is configured as open drain.
20. Fake fault cannot be generated for FCCU channel (requires 50 us delay to be active).

7.13.5 STCU2 configuration

7.13.5.1 Wait time for writing to the online registers

When writing to the online registers, the user must wait for the expiration of the offline key. The time the user should wait is (IPS_CLK)*4096 clock periods. If the IPS_CLK is 66.5 Mhz then the user must wait (1/66.5)*4096 = 61.6 µs.

7.13.6 Register protection (REG_PROT) configuration

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the module. This mechanism is software-controlled and can be dynamically reconfigured at runtime or made effective until device reset. For a detailed explanation, see the Register Protection (REG_PROT).

Table 7-87. Reference links to related information

Topic	Related module	Reference
System memory map	—	Memory Map
SIUL2 registers	SIUL2	System Integration Unit Lite2 (SIUL2)
CMU registers	CMU, CMUIOP	Clock Monitor Unit (CMU)
PMC Digital Interface (PMC_DIG) registers	PMC_DIG	Power Management Controller digital interface (PMC_dig)
PLL Digital Interface (PLL_DIG) registers	PLL_DIG	Dual PLL Digital Interface (PLLDIG)
Oscillator Digital Interface (OSC_DIG) registers	OSC_DIG	OSC Digital Interface (XOSC)
Memory Error Management Unit (MEMU) registers	MEMU	Memory Error Management Unit (MEMU)
Mode Entry (MC_ME) registers	MC_ME	Mode Entry Module (MC_ME)
Clock Generation Module (ME_CGM) registers	MC_CGM	Clock Generation Module (MC_CGM)
Reset Generation Module (MC_RGM) registers	MC_RGM	Reset Generation Module (MC_RGM)
System Status and Control Module (SSCM) registers	SSCM	System Status and Configuration Module (SSCM)
Indirect Memory Access (IMA) registers	IMA	Indirect Memory Access (IMA)">Indirect Memory Access (IMA)
JTAG Master (JTAGM) registers	JTAGM	JTAG Master (JTAGM)

7.13.6.1 SIUL2 protected registers

Table 7-88 lists the SIUL2 registers that can be protected.

Table 7-88. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SIUL2_DIRER0	32	0x0018	32
SIUL2_DIRSR0	32	0x0020	32
SIUL2_IREER0	32	0x0028	32
SIUL2_IFEER0	32	0x0030	32
MSCR 0–4	32	0x0240	32 (×5)
MSCR 8–15	32	—[1]	32 (×8)
MSCR 24–27	32	—[1]	32 (×4)
MSCR 32–47	32	—[1]	32
MSCR 49	32	—[1]	32
MSCR 51–54	32	—[1]	32
MSCR 56–76	32	—[1]	32
MSCR 80–89	32	—[1]	32
MSCR 91	32	—[1]	32
MSCR 109–127	32	—[1]	32
MSCR 136–159	32	—[1]	32
MSCR 172–202	32	—[1]	32
MSCR 512-519	32	—[1]	32
MSCR 520-527	32	—[1]	32
MSCR 528-535	32	—[1]	32
MSCR 536-543	32	—[1]	32
MSCR 576-577	32	—[1]	32
MSCR 578-579	32	—[1]	32
MSCR 580-581	32	—[1]	32
MSCR 582-583	32	—[1]	32
MSCR 584-585	32	—[1]	32
MSCR 586-587	32	—[1]	32
MSCR 588-589	32	—[1]	32
MSCR 590-591	32	—[1]	32
MSCR 608	32	—[1]	32
MSCR 609	32	—[1]	32
MSCR 610	32	—[1]	32
MSCR 611	32	—[1]	32
MSCR 612	32	—[1]	32
MSCR 614	32	—[1]	32

Table continues on the next page...

Table 7-88. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MSCR 615	32	—[1]	32
MSCR 616	32	—[1]	32
MSCR 617	32	—[1]	32
MSCR 618	32	—[1]	32
MSCR 624	32	—[1]	32
MSCR 625	32	—[1]	32
MSCR 626	32	—[1]	32
MSCR 627	32	—[1]	32
MSCR 628	32	—[1]	32
MSCR 630	32	—[1]	32
MSCR 631	32	—[1]	32
MSCR 632	32	—[1]	32
MSCR 633	32	—[1]	32
MSCR 634	32	—[1]	32
MSCR 638	32	—[1]	32
MSCR 639	32	—[1]	32
MSCR 640	32	—[1]	32
MSCR 641	32	—[1]	32
MSCR 642	32	—[1]	32
MSCR 643	32	—[1]	32
MSCR 644	32	—[1]	32
MSCR 645	32	—[1]	32
MSCR 646	32	—[1]	32
MSCR 647	32	—[1]	32
MSCR 648	32	—[1]	32
MSCR 649	32	—[1]	32
MSCR 656-687	32	—[1]	32
MSCR 688-719	32	—[1]	32
MSCR 720-751	32	—[1]	32
MSCR 752-754	32	—[1]	32
MSCR 758	32	—[1]	32
MSCR 759	32	—[1]	32
MSCR 760	32	—[1]	32
MSCR 761	32	—[1]	32
SIUL2_GPDO0–340	8	0x1300	8 (×341)
SIUL2_PGPDO0–21	16	0x1700	16 (×22)
SIUL2_MPGPDO0–21	32	0x1780	32 (×22)

1. See the module memory map in the SIUL2 chapter for address offset.

7.13.6.2 Core Clock Monitor Unit (CMU_CORE) protected registers

Table 7-89 lists the registers that can be protected for the core CMU.

Table 7-89. Protected CMU_CORE registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_CSR	32	0x0	8 (byte 0)

7.13.6.3 I/O Processor Clock Monitor Unit (CMU_IOP) protected registers

Table 7-90 lists the registers that can be protected for the I/O Processor CMU.

Table 7-90. Protected CMU_IOP registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_CSR	32	0x0	8 (byte 0)

7.13.6.4 PMC Digital Interface (PMC_DIG) protected registers

Table 7-91 lists the PMC_DIG registers can be protected.

Table 7-91. Protected PMC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_VD3	32	0x0034	32
RES_VD3	32	0x0038	32
FEE_VD3	32	0x003C	32
REE_VD4	32	0x0044	32
RES_VD4	32	0x0048	32
FEE_VD4	32	0x004C	32
REE_VD7	32	0x0074	32
RES_VD7	32	0x0078	32
FEE_VD7	32	0x007C	32
REE_VD9	32	0x0094	32
RES_VD9	32	0x098	32

Table continues on the next page...

Table 7-91. Protected PMC_DIG registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
FEE_VD9	32	0x009C	32
REE_VD10	32	0x0A4	32
FEE_VD10	32	0x00AC	32
REE_VD12	32	0x00C4	32
RES_VD12	32	0x00C8	32
FEE_VD12	32	0x00CC	32
REE_VD13	32	0x00D4	32
RES_VD13	32	0x00D8	32
FEE_VD13	32	0x00DC	32
REE_VD14	32	0x00E4	32
RES_VD14	32	0x00E8	32
FEE_VD14	32	0x00EC	32
REE_VD15	32	0x00F4	32
RES_VD15	32	0x00F8	32
FEE_VD15	32	0x00FC	32
REE_TD	32	0x0304	32
RES_TD	32	0x0308	32
CTL_TD	32	0x030C	32
FEE_TD	32	0x0318	32
VD_UTST	32	0x340	8 (byte 0)
ADC_CH	32	0x344	8 (byte 0)

7.13.6.5 PLL Digital Interface (PLL_DIG) protected registers

Table 7-92 lists the PLL_DIG registers that can be protected.

Table 7-92. Protected PLL_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR — PLL0 Control Register	32	0x0000	32
PLL0_DIVR — PLL0 Divider Register	32	0x0008	32
PLL1_CR — PLL1 Control Register	32	0x0020	32
PLL1_DIVR — PLL1 Divider Register	32	0x0028	32
PLL1_FMR — PLL1 Frequency Modulation Register	32	0x002C	32
PLL1_FDR — PLL1 Fractional Divide Register	32	0x0030	32

7.13.6.6 Oscillator Digital Interface (OSC_DIG) protected registers

Table 7-93 lists the OSC_DIG registers that can be protected.

Table 7-93. Protected OSC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
OSC_CTL	32	0x0000	32

7.13.6.7 Memory Error Management Unit (MEMU) protected registers

Table 7-94 lists the MEMU registers that can be protected.

Table 7-94. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	32	0x000	8 (byte 3)
DEBUG	32	0x00C	32

7.13.6.8 Mode Entry (MC_ME) protected registers

Table 7-95 lists the MC_ME registers that can be protected.

Table 7-95. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_MCTL	32	0x0004	32
ME_ME	32	0x008	32
ME_IM	32	0x010	32
ME_SAFE_MC	32	0x028	32
ME_DRUN_MC	32	0x02C	32
ME_RUN0_MC	32	0x030	32
ME_RUN1_MC	32	0x034	32
ME_RUN2_MC	32	0x038	32
ME_RUN3_MC	32	0x03C	32
ME_RUN_PC0–7	32	0x080–0x09C	32
ME_PCTL 0–255 [1]	8	0x0C4–0x1BF	8
ME_CCTL0–4	16	0x1C4–0x1CC	16

1. Only valid PCTLs [3,9,11,15,30,31,36,38,56-60,67-70,72,74,84,85,91-93,96-99,101,104,107,111,112,123,127,128,162,166,184-188,213,220,225-227,229,232,235,239,245-250,252-254] are register protected. See the module memory map in the MC_ME chapter for address offsets.

7.13.6.9 Clock Generation Module (MC_CGM) protected registers

Table 7-96 lists the MC_CGM registers that can be protected.

Table 7-96. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_SC_DC[0..3]	32	0x07E8–0x07F4	32
CGM_AC0_SC	32	0x0800	8 (byte 3)
CGM_AC1_SC	32	0x0820	8 (byte 3)
CGM_AC3_SC	32	0x0860	8 (byte 3
CGM_AC4_SC	32	0x0880	8 (byte 3
CGM_AC[6..10]_SC	32	0x08C0–0x0940	8 (byte 3)
CGM_AC[0..15]_DC[0..3]	32	0x0808..0814–0x09E8..09F4	32

7.13.6.10 Reset Generation Module (MC_RGM) protected registers

Table 7-97 lists the MC_RGM registers that can be protected.

Table 7-97. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RGM_DERD	32	0x010	32
RGM_DEAR	32	0x020	32
RGM_DBRE	32	0x030	32
RGM_FERD	32	0x310	32
RGM_FEAR	32	0x320	32
RGM_FBRE	32	0x330	32
RGM_FESS	32	0x340	32
RGM_PRSTn	32	0x610+n*0x4[1]	32

1. n = 0,1, 2, 3, 4, 5, 6, 7

7.13.6.11 System Status and Control Module (SSCM) protected registers

Table 7-98 lists the SSCM registers that can be protected.

Table 7-98. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ERROR	16	0x0006	16

7.13.6.12 Indirect Memory Access (IMA) protected registers

Table 7-99 lists the IMA registers that can be protected.

Table 7-99. Protected IMA registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
IMA_ENABLE	32	0x0004	32

7.13.6.13 JTAG Master (JTAGM) protected registers

Table 7-100 lists the JTAGM registers that can be protected.

Table 7-100. Protected JTAGM registers

Register	Register Size (bits)	Offset from Module Base Address	Protected Size (bits)
JTAGM_MCR	32	0x00	32

7.13.7 Security modules

The modules described in this section provide a basic set of customer-configurable security features designed to protect code and data from unauthorized access and to provide a way implement an audit trail to identify accesses to specific blocks of flash memory.

Table 7-101. Reference links to related information

Topic	Related module	Reference
Security overview	—	Security
Password protection of flash memory blocks	PASS	Flash Memory Programming and Configuration
	PASS	Password and Device Security Module (PASS)
Tamper Detection	TDM	Tamper Detection Module (TDM)
Detailed information on advanced security features	—	See the Microcontroller Security Reference Manual

7.13.8 Password and Device Security Module (PASS) configuration

The PASS module is used to implement password-based read and write protection for flash blocks on the MCU. Up to 4 levels of password protection can be implemented for each flash block.

Note

The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See the Password and Device Security Module (PASS) chapter for further details.

7.13.8.1 PASS_LOCK*n*_PGn register bit mapping

As described in Flash Memory Programming and Configuration, each password group defined in the DCF records has a set of 4 LOCKn registers association with it—PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the following figures.

Note

The mapping of various LOCK register bits to flash blocks in the MPC5777M microcontroller is very similar, but not identical. The PASS module has additional mapping in the LOCK3 register and defines read locking regions. See the PASS chapter for details.

Refer to Secure read protection for details of flash read lock protection.

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register Field	TSLOCK	ATSL	LOWLOCK[13:0]														MIDLOCK[15:0]															
Flash Block Name	UTest NVM Block Space 16 KB	—	—	—	64 KB HSM Code block3	64 KB HSM Code block2	64 KB Code Flash block1	64 KB Code Flash block0	32 KB Code Flash block1	32 KB Code Flash block0	16 KB HSM Code block5	16 KB Code Flash block4	16 KB Code Flash block3	16 KB Code Flash block2	16 KB Code Flash block1	16 KB BAF block0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB HSM EEPROM block1	16 KB HSM EEPROM block0
Read Lock Region	Read Lock Region 0				Read Lock Region 3			Read Lock Region 1			Read Lock Region 3		Read Lock Region 1																	Read Lock Region 4	

Figure 7-26. PASS_LOCK0_PG*n* Register

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register Bit Name	ATSL	Reserved															HIGHLOCK[15:0]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	EEPROM Block7	EEPROM Block6	EEPROM Block5	EEPROM Block4	EEPROM Block3	EEPROM Block2	EEPROM Block1	EEPROM Block0
Read Lock Region	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Read Lock Region 2							

Figure 7-27. PASS_LOCK1_PG*n* Register

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register Bit Name	256LCK_L[31:0]																															
Flash block Name	—	—	256KB Code Flash block29	256KB Code Flash block28	256KB Code Flash block27	256KB Code Flash block26	256KB Code Flash block25	256KB Code Flash block24	256KB Code Flash block23	256KB Code Flash block22	256KB Code Flash block21	256KB Code Flash block20	256KB Code Flash block19	256KB Code Flash block18	256KB Code Flash block17	256KB Code Flash block16	256KB Code Flash block15	256KB Code Flash block14	256KB Code Flash block13	256KB Code Flash block12	256KB Code Flash block11	256KB Code Flash block10	256KB Code Flash block9	256KB Code Flash block8	256KB Code Flash block7	256KB Code Flash block6	256KB Code Flash block5	256KB Code Flash block4	256KB Code Flash block3	256KB Code Flash block2	256KB Code Flash block1	256KB Code Flash block0
Read Lock Region	—	—	Read Lock Region 1																													

Figure 7-28. PASS_LOCK2_PG*n* Register

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register Bit Name	PGL	DBL	MO	Reserved	MSTR				Reserved			RL4	RL3	RL2	RL1	RL0	256KLOCK_U[15:0]															
Flash block Name	Password Group Lock	Debug Interface Lock	Master Only	—	Master Access				—	—	—	HSM EEPROM Data Flash Read Lock	HSM Code Flash Read Lock	EEPROM Data Flash Read Lock	Code Flash Read Lock	UTEST Read Lock	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 7-29. PASS_LOCK3_PG*n* Register

7.13.8.2 PASS Module DCF Records

The records required to configure the initial values for the PASS module password group registers is shown in Table 7-102. These values provide the following capabilities:

- Set the current life cycle state
- Set flash memory to censored/uncensored state (censorship also depends on lifecycle)
- Override the basic flash write protection for flash memory blocks
- Set levels of password-based write protection (up to four 256-bit passwords can be required) for individual blocks of flash memory
- Set levels of password-based read protection (up to four 256-bit passwords can be required) for up to five flash memory Read Locking Regions
- Set levels of password-based protection (up to four 256-bit passwords can be required) for the debug part.

Most DCF records represent an initial value or reset value for a module register. See the relevant register documentation for details on values that can be assigned.

Table 7-102. PASS DCF records

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0000_1000	0000000	00101001–	Reserved		

Table continues on the next page...

Table 7-102. PASS DCF records (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
		00101011			
000_0000_0000_1000	0000000	00101100	Censorship[1]	16	1
000_0000_0000_1000	0000000	00101101–00101111	Reserved		
000_0000_0000_1000	0000000	00110000	Production Disable	2	1
000_0000_0000_1000	0000000	00110001–00111111	Reserved		
000_0000_0000_1000	0000000	01000000	LOCK0_PG0[-1]	32	1
000_0000_0000_1000	0000000	01000001	LOCK1_PG0[3]	32	1
000_0000_0000_1000	0000000	01000010	LOCK2_PG0[3]	32	1
000_0000_0000_1000	0000000	01000011	LOCK3_PG0[3]	32	1
000_0000_0000_1000	0000000	01000100	LOCK0_PG1[3]	32	1
000_0000_0000_1000	0000000	01000101	LOCK1_PG1[3]	32	1
000_0000_0000_1000	0000000	01000110	LOCK2_PG1[3]	32	1
000_0000_0000_1000	0000000	01000111	LOCK3_PG1[3]	32	1
000_0000_0000_1000	0000000	01001000	LOCK0_PG2[3]	32	1
000_0000_0000_1000	0000000	01001001	LOCK1_PG2id-73121	32	1
000_0000_0000_1000	0000000	01001010	LOCK2_PG2[3]	32	1
000_0000_0000_1000	0000000	01001011	LOCK3_PG2[3]	32	1
000_0000_0000_1000	0000000	01001100	LOCK0_PG3[3]	32	1
000_0000_0000_1000	0000000	01001101	LOCK1_PG3[3]	32	1
000_0000_0000_1000	0000000	01001110	LOCK2_PG3[3]	32	1
000_0000_0000_1000	0000000	01001111	LOCK3_PG3[3]	32	1
000_0000_0000_1000	0000000	01001100–11111111	Reserved		

1. See Censoring and uncensoring the device for values.
2. See External Bus Interface (EBI) Support for values.
3. See External Bus Interface (EBI) Support for values.

7.13.9 Tamper Detection Module (TDM) configuration

The Tamper Detection Module provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

Additionally, the TDM provides a mechanism for configuring individual flash memory blocks as One Time Programmable (OTP), i.e., write once, blocks.

The following sections provide details on the TDM configuration registers and DCF clients.

Warning

The TDM configuration registers detailed in this section are DCF clients. In general, DCF clients can only be written once —their intended values must be carefully selected beforehand.

See the Tamper Detection Module (TDM) chapter for further details on the TDM, including the DCF clients and two additional memory-mapped registers.

7.13.9.1 Diary Base Address (DBA) DCF client

The TDM_DBA register holds the base address of the diary, which is the region of flash that contains records corresponding to erase operations. This register is more fully described in the TDM chapter.

7.13.9.2 Tamper Region Override (TO) DCF client

Tamper detection can be overridden by writing to enable bits (one per region) in the Tamper Region Override (TO) DCF client. This DCF client is more fully described in the TDM chapter. The mapping is shown below.

Offset: 0x01 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-30. Tamper Region Override (TO) DCF client

The mapping of the Tamper Region Override Enable (TOE*n*) is as follows:

Table 7-103. _TDM_TO[TOEn] field mapping

Field	TDR
TOE0	0
TOE1	1
TOE2	2

Table continues on the next page...

Table 7-103. _TDM_TO[TOEn] field mapping (continued)

Field	TDR
TOE3	3
TOE4	4
TOE5	5

This DCF client is more fully described in the TDM chapter

7.13.9.3 OTPEN*n* DCF clients

Configuring individual flash blocks as OTP blocks is accomplished by setting the appropriate values in the TDM_OTPEN*n* DCF clients.Each mapped bit is associated with a specific flash block. A "1" bit value indicates a block is OTP.

The mapping is shown in Figure 7-31 - Figure 7-34.

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved		LOWLOCK[13:0]														MIDLOCK[15:0]															
Flash Block Name	—	—	—	—	64 KB HSM Code block3	64 KB HSM Code block2	64 KB Code Flash block1	64 KB Code Flash block0	32 KB Code Flash block1	32 KB Code Flash block0	16 KB HSM Code block5	16 KB Code Flash block4	16 KB Code Flash block3	16 KB Code Flash block2	16 KB Code Flash block1	16 KB BAF block0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB HSM EEPROM block1	16 KB HSM EEPROM block0

Figure 7-31. TDRx_LOCK0/OTPEN0 DCF client

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	Reserved																HIGHLOCK[15:0]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	EEPROM Block7 (64KB EEPROM)	EEPROM Block6 (64KB EEPROM)	EEPROM Block5 (64KB EEPROM)	EEPROM Block4 (64KB EEPROM)	EEPROM Block3 (64KB EEPROM)	EEPROM Block2 (64KB EEPROM)	EEPROM Block1 (64KB EEPROM)	EEPROM Block0 (64KB EEPROM)

Figure 7-32. TDRx_LOCK1/OTPEN1 DCF client

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	256KLOCK[31:0]																															
Flash block Name	—	—	256KB Code Flash block29	256KB Code Flash block28	256KB Code Flash block27	256KB Code Flash block26	256KB Code Flash block25	256KB Code Flash block24	256KB Code Flash block23	256KB Code Flash block22	256KB Code Flash block21	256KB Code Flash block20	256KB Code Flash block19	256KB Code Flash block18	256KB Code Flash block17	256KB Code Flash block16	256KB Code Flash block15	256KB Code Flash block14	256KB Code Flash block13	256KB Code Flash block12	256KB Code Flash block11	256KB Code Flash block10	256KB Code Flash block9	256KB Code Flash block8	256KB Code Flash block7	256KB Code Flash block6	256KB Code Flash block5	256KB Code Flash block4	256KB Code Flash block3	256KB Code Flash block2	256KB Code Flash block1	256KB Code Flash block0

Figure 7-33. TDR*x*_LOCK2/OTPEN2 DCF client

Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	Reserved																256KLOCK[47:32]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 7-34. TDR*x*_LOCK3/OTPEN3 DCF client

7.13.9.4 TDR*x*_LOCK*n* DCF clients

Configuration of the 6 tamper regions is accomplished by setting field values in LOCK DCF clients (4 for each TDR).

Each set of 4 LOCKn DCF clients acts as a map of the MCU's flash memory blocks. A "1" in a LOCKn bit indicates that software must write a record before the corresponding block can be modified. The mapping of TDM_LOCKn DCF client bits to flash blocks is identical to the OTPEN*n* mapping and is shown in Figure 7-31 - Figure 7-34. Any available flash block can be mapped to any tamper protection region.

7.13.9.5 DCF client details

Table 7-104 contains the DCF client information.

Table 7-104. Tamper Detection Module (TDM) DCF clients

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00000000	Diary Base Address	32	1
000_0000_0001_0000	0000000	00000001	Tamper Region Override	6	1

Table continues on the next page...

Table 7-104. Tamper Detection Module (TDM) DCF clients (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00000010 - 00000111	Reserved		
000_0000_0001_0000	0000000	00001000	OTPEN0[1]	15	1
000_0000_0001_0000	0000000	00001001	OTPEN1[1]	4	1
000_0000_0001_0000	0000000	00001010	OTPEN2[1]	14	1
000_0000_0001_0000	0000000	00001011	OTPEN3[2]	0	1
000_0000_0001_0000	0000000	00001100- 00010011	Reserved		
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	32	1
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	32	1
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	32	1
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	32	1
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	32	1
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	32	1
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	32	1
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	32	1
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	32	1
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	32	1
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	32	1
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	32	1
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	32	1
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	32	1
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	32	1
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	32	1
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	32	1

Table continues on the next page...

Table 7-104. Tamper Detection Module (TDM) DCF clients (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	32	1
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	32	1
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	32	1
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	32	1
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	32	1
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	32	1
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	32	1
000_0000_0001_0000	0000000	00101100 - 1111111	Reserved		

1. OTP DCF clients have the same structure and mapping as the write-protect bits of the flash and PASS module LOCK DCF clients. A value of '1' in an OTP bit indicates a block is programmable only once. Note that a block cannot be reconfigured as non-OTP after it has been configured as OTP.
2. There are no flash memory blocks mapped to the OTPEN3 DCF client.

7.13.10 Production device vs. emulation device

MPC5777M microcontrollers have a comprehensive feature set to assist debug and calibration. Specialized variants, referred to as Emulation Devices (EDs), include additional calibration and debug features. An ED chip consists of the normal production die plus an additional die, called a Buddy die or Buddy Device (BD), bonded together in the same chip package. A Production Device (PD) includes a production die only and no buddy die.

Note

All features and modules included in the PD are also included in the ED.

The following table lists the key differences between the ED and PD of the MPC5777M microcontroller.

Table 7-105. Debug and calibration features per MPC5777M type

Debug or calibration feature	MPC5777M PD	MPC5777M ED
Internal Overlay Memory	16 KB	16 KB
Extended Overlay Memory	—	2 MB
Debug Interface	JTAG / LFAST	JTAG / LFAST
Trace Interface	—	Aurora (1.25 GHz, Lane)
Trace Sources	Data trace on all bus accesses initiated from PD, Program trace on main CPUs, DMA and GTM	Data trace on all bus accesses initiated from PD, Program trace on main CPUs, DMA and GTM
Trace Destination	Internal Overlay	Internal Overlay, Extended Overlay, Aurora Trace Port
Calibration Remap capabilities	32 Region	32 Region
System Level Performance Monitor Counters[1]	16 × 32 bits	16 × 32 bits
CPU Level Performance Monitor Counters	12 × 32 bits (4 × 32 bits per z4 CPU)	12 × 32 bits (4 × 32 bits per e200z4 CPU)

1. All system level PMCs can also be configured for use as debug timers. System level PMCs can also be configured as timers.

The following figure shows the top-level block diagram of the MPC5777M ED, which consists of the production die packaged together with the 2 MB Buddy die (BD2M). The figure shows the general interconnection of functional debug modules including interconnections between the two devices. All functional blocks are shown for the BD, but for clarity only blocks relevant to the debug and calibration system are shown for the PD.

Acronyms used in the block diagram include:

- BD_SIU — BD System Integration logic
- BD_XBS_RAM — Buddy Device RAM Crossbar
- DCI — Debug and Calibration Interface
  - JTAGC — JTAG Controller
  - M.JTAG — Mastered JTAG
  - P.JTAG — Peripheral JTAG
  - I.JTAG — Inter-die JTAG
  - DCU — Debug Control Unit
- DTS — Development Tool Semaphore

- JTAGM — JTAG Master
- SPU — Sequence Processing Unit
- NAR — Nexus Aurora Router
- NAL — Nexus Aurora Link
- Nexus RWA — Nexus Read/Write Access client

On ED devices, the device packaging routes the tool interface to the interface logic on the BD. On standard production devices packages the device packaging routes the tool interface to the interface logic on the PD.

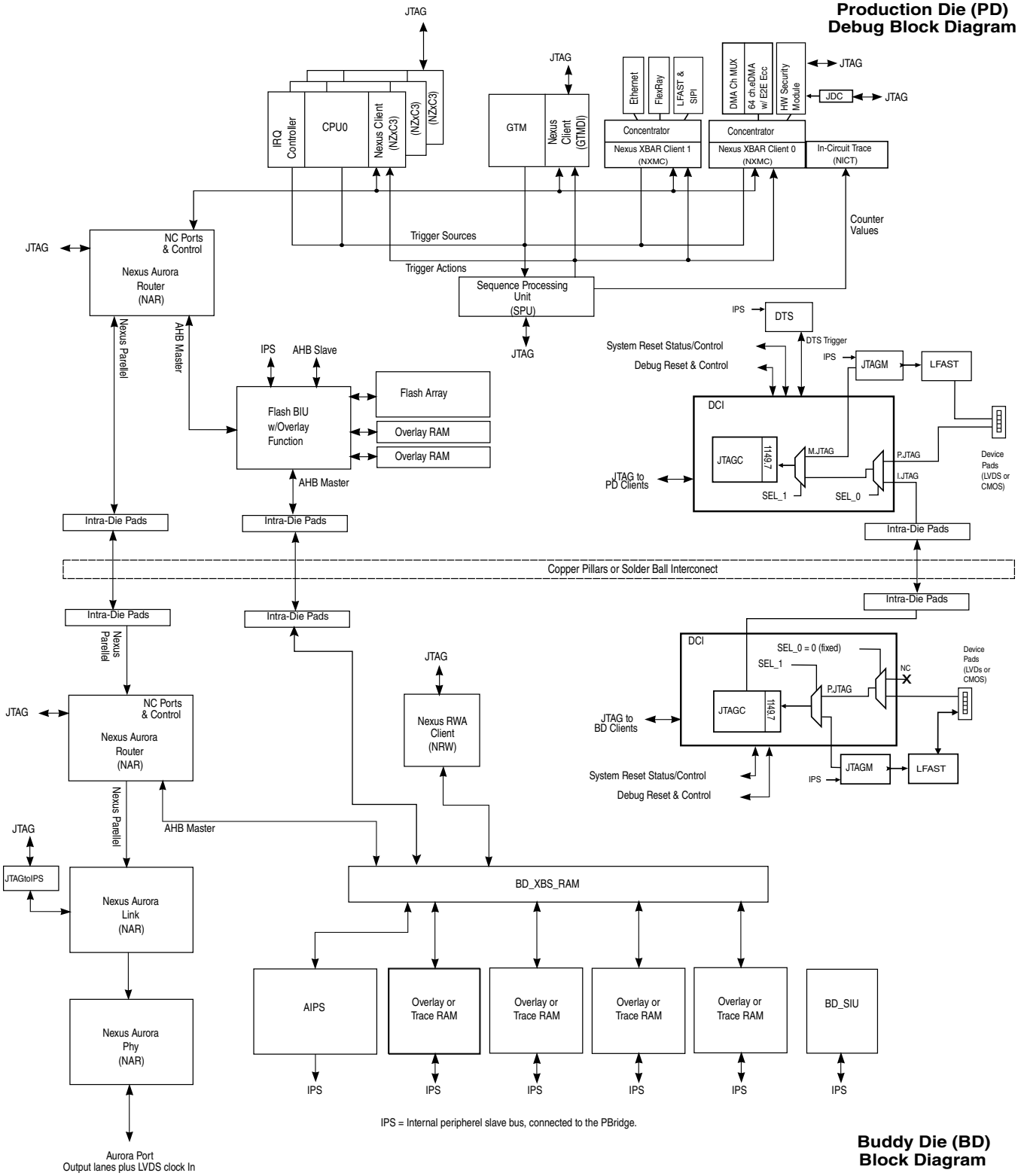


Figure 7-35. Debug and calibration architecture for MPC5777M ED

Chapter 8
Reset and Boot

8.1 Introduction

Resetting the MPC5777M starts with the application of power and ends with fetching the first instruction of the user's application code. A review of chip architecture is helpful to appreciate reset.

The processor cores on the MPC5777M are:

- Core_0 and Core_1 based on the e200_z710 architecture, generally execute user application code.
- Checker Core_0 executes the same instruction stream as Core_0. The results of each instruction execution are compared between the two cores and mismatches are flagged.
- Peripheral Core_2 (IOP or I/O Processor) is implemented using an e200_z425 core. It is intended to control the various peripherals on the chip, but may be used to execute any user code.
- A Hardware Security Module (HSM) with an embedded e200_z0 core is also included. The HSM executes factory-written code stored in Secure Flash only accessible to the HSM. Like the other processor cores, the HSM can access other memory elements on the chip.

Memory elements involved in the boot-up process include:

- UTEST flash memory
- TEST flash memory
- Boot Assist Flash (BAF).

8.1.1 Boot Assist Flash

The Boot Assist Flash (BAF) contains factory code to facilitate the boot-up procedure.

If the BAF code does not locate a valid boot header, it runs the Serial Boot Loader which receives files from an external serial link controlled by the internal LINFlexD module. The received file is placed in RAM and program execution begins at the start address specified in the received file.

The Serial Boot Loader is only an option in the early stages of the device life cycle. In later stages (for example, FIELD) the device waits for the watchdog to reset the device if no boot header is found.

8.1.2 TEST flash memory block

The TEST flash memory block contains Device Configuration Format (DCF) records used to hold trim values and other variables, as well as general device configuration information.

The trim values are for:

- Adjusting low-voltage and high-voltage detect circuit trip points
- Temperature sensor adjustments
- Power-on reset voltage trip point
- Analog-to-digital adjustments
- Internal RC oscillator (IRCOSC) trim values

NOTE

The DCF records are written by the device manufacturer and programmed into TEST flash memory during production testing. Further programming of the TEST flash is disabled at the end of the factory test cycle.

8.1.3 UTEST flash memory block

The UTEST flash memory block also contains DCF records. Some UTEST DCF records are written by the factory and programmed during production testing. Others are written by the end user and programmed at the same time application code is programmed into the flash memory. UTEST DCF records are used to set up various control and configuration registers, including the Self-Test Control Unit (STCU2) and default memory configuration.

8.1.4 Boot header

A boot header is used to supply the start address for code execution for Core_0, Core_1 and Checker Core_0s. It is written by the application programmer and loaded into specific flash memory locations. It contains the reset vectors for the various cores and specifies the BOOT CPU to execute application code.

8.2 Modules used in reset sequence

The modules involved in the reset sequence are:

- Power Management Controller (PMC)
- Reset Generation Module (MC_RGM)
- Mode Entry Module (MC_ME)
- System Status and Configuration Module (SSCM)
- Self-Test Control Unit (STCU2)

8.2.1 Power Management Controller

The Power Management Controller (PMC) controls and monitors:

- Its own supply voltage
- The supply voltages to all the high- and low-voltage detect circuits
- The trip points for all the high- and low-voltage detect circuits
- The power supplies and reference voltages to the Analog-to-Digital Converter
- The major power supplies to the chip

8.2.2 Reset Generation Module

The Reset Generation Module (MC_RGM) is a complex state machine that begins sequencing the chip through the initial steps of the reset process. The MC_RGM does not execute program code, it is a state machine that centralizes the different reset sources and manages the reset sequence. Reset sources are organized into two categories: destructive and functional.

See Figure 8-1 for more information on the MC_RGM reset sequence.

8.2.2.1 Destructive resets

A destructive reset source is related to a critical error or dysfunction, usually caused by a hardware malfunction. When a destructive reset event occurs, software recovery is not possible and the contents of memory are assumed unknown. A full device reset sequence starting from PHASE0 is therefore applied to the device, ensuring a safe startup state for both digital and analog modules.

Examples of destructive resets are:

- Power-on reset
- Low voltage detection
- Reset escalation (when too many functional resets occur simultaneously)

8.2.2.2 Functional resets

A functional reset source is related to a less critical error or dysfunction not usually not associated with hardware malfunction. When a functional reset event occurs, a partial reset sequence is applied to the device starting with PHASE1[FUNC]: most digital modules are reset normally and the state of analog modules, specific digital modules (for example, debug and flash memory modules) and system memory content is preserved.

Examples of functional resets are:

- External reset
- Machine check
- Software reset from mode entry
- Boundary scan instructions

8.2.3 Mode Entry module

The Mode Entry module (MC_ME) is responsible for delivering initial values to many registers, including the reset vectors to all the cores.

During the power-up sequence, the System Status and Configuration Module (SSCM) searches for a boot header at defined locations in the flash memory, derives the necessary reset vectors from the boot header and then writes these vectors to their respective locations in the Mode Entry module.

The SSCM also interprets the DCF records which contain further information that is written to the MC_ME. As the boot-up sequence progresses, the SSCM instructs the MC_ME to transfer the reset vectors to the respective processor cores.

Even though the reset vectors are in the boot header located in flash memory, the Mode Entry module always provides them to the CPU cores.

8.2.4 System Status and Configuration Module (SSCM)

During the reset sequence, the Reset Generation Module (MC_RGM) enables the SSCM state machine which reads the DCF records in the TEST and UTEST flash memory areas for device setup information to be written to various registers in the chip.

The DCF records are primarily for setting up the memory, configuring the Self-Test Control Unit (STCU2), and providing initial device configuration values.

The SSCM continues with the reset or boot-up sequence by locating boot vectors for the various cores. Once the SSCM locates the boot header, the SSCM writes the start address information for the CPU cores to special locations in the Mode Entry Module (MC_ME). For all cores enabled by the boot header, the MC_ME feeds a reset vector to each enabled core and the respective cores begins program execution at the specified address.

The SSCM starts up the IOP and the HSM in different ways depending on whether there is valid code in the Boot Assist Flash (BAF).

The boot-up sequence ends when the SSCM starts up the Core_0 or Core_1 boot CPU.

8.2.5 Self-Test Control Unit (STCU2)

The Self-Test Control Unit (STCU2) is a self-contained module that runs a Logic Built-In Self Test (LBIST) and a Memory Built-In Self Test (MBIST). The DCF record can be used to select which individual tests are run for LBIST and MBIST and disable tests for modules and memory elements that are not going to be used in user applications, thus shortening device boot-up time.

8.3 Reset sequence

The power-up reset sequence always begins with the application of power and follows different sequences depending on the condition of the MPC5777M chip and whether various modes are enabled from settings in the DCF records. For instance, the chip enters serial boot mode (the Serial Boot Loader receives startup code and begins program execution) if a valid boot header file is not found.

If an emulation and debug device is detected during the reset sequence, the device enters the calibration sequence. The calibration sequence is not a user mode and is only used for code development.

Another reset outcome is that the Hardware Security Module, the IO Processor, and the boot CPU are all properly started and begin executing their respective program code.

8.3.1 Power-on and the Reset Generation Module

When power is applied to the chip, the Reset Generation Module (MC_RGM) advances the device through a series of steps shown in Figure 8-1.

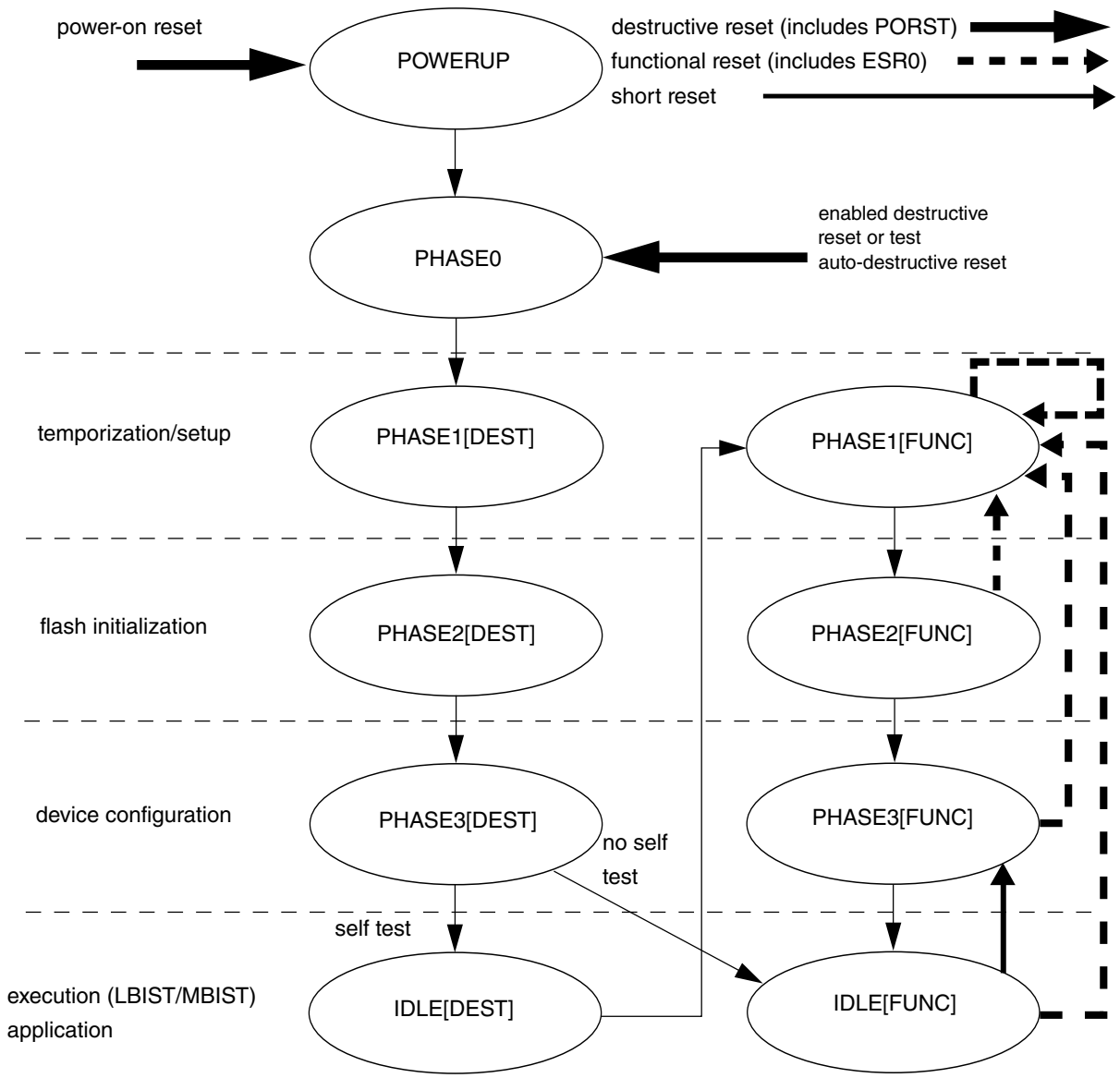


Figure 8-1. Reset Generation Module Reset Sequence

The MC_RGM starts the System Status and Control Module (SSCM), which continues the boot-up process detailed in Figure 8-2 through to Figure 8-6 after the MC_RGM enters the IDLE state.

The STCU is reset on any power-on, 'destructive,' or long external reset (external reset in this case is ESR0).

Table 8-1. Module status during reset phases

Module	PHASE								
	P0	P1 [DEST]	P2 [DEST]	P3 [DEST]	IDLE [DEST]	P1 [FUNC]	P2 [FUNC]	P3 [FUNC]	IDLE [FUNC]
IRCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	ON[1]	ON	ON	ON	ON	ON	ON	ON	ON
System watchdog	RESET	RESET	ON	RESET	LBIST[2]	RESET	ON	RESET	ON[3]
Flash module	RESET	RESET	ON	ON	LBIST	RESET	ON	ON	ON
XOSC	RESET	RESET	RESET	ON[3]	ON[3]	ON[3]	ON[3]	ON[3]	ON[3]
DEVICE CONFIG	RESET	RESET	RESET	LOAD	HOLD	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	LBIST	RESET/ON[4]	CONFIG/ON	CONFIG /ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	RESET/ IDLE[5]	CONFIG/ IDLE	CONFIG /IDLE	ON/ IDLE[6]
MEMU	RESET	RESET	ON	ON	LBIST	RESET/ON[7]	ON	ON	ON
PASS	RESET	ON	ON	ON	ON	ON	ON	ON	ON
HSM	RESET	RESET	RESET	RESET	LBIST[8]	RESET	RESET	RESET	ON
IOP	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON
PIT_1	RESET	RESET	RESET	RESET	LBIST	IDLE/ON	IDLE/ON	IDLE/O N	IDLE/O N
Other modules including cores	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON[9]
								Short functional reset	
						functional reset			
	destructive reset								

1. RESET only during POR
2. STCU time-out monitoring LBIST/MBIST execution
3. Depending on flash configuration
4. FCCU remains ON during functional reset; it is instead reset in PHASE1[FUNC] if coming from IDLE[DEST]. FCCU is reset whenever the STCU is reset plus the STCU's post self-test functional reset request plus the individual peripheral reset from the RGM_PRSTn registers.
5. The STCU remains idle during functional reset; it is instead reset on long external reset (external reset here is ESR0).

6. Can be triggered by SW. Please refer to the STCU2 chapter.
7. MEMU is reset whenever STCU is reset.
8. LBIST is optionally triggered depending on safety execution directives
9. LBIST is optionally triggered depending on safety execution directives after enabling from IOP. Please refer to the BAF chapter.

8.3.2 Power-up phase: power stabilization

When a power-on reset event occurs (initial application of power), the reset generation module (MC_RGM) causes the device to enter the power-up phase of the reset sequence (See Figure 8-1). The only method to enter the power-up phase is from a power-on reset event.

The device remains in the power-up phase until the Power Management Controller module (PMC) determines that all power supplies are in their respective operational ranges.

The supplies for initial configuration are:

- VDD_HV_CORE: high voltage supply for core module, including power management unit, internal RC oscillator (IRCOSC), external oscillator (XOSC) driver
- VDD_LV_CORE: low voltage supply for digital module
- VDD_HV_FLA: high voltage supply for flash module
- VDD_LV_FLA: low voltage supply for flash module
- VDD_HV_IO_IM: high voltage supply associated with system pins (PORST, ESR0, TEST Mode)

Peripheral supplies, such as VDD_HV_IO_JTAG, VDD_HV_IO_FLEX, or VDD_HV_ADC, do not gate the power-up process. The low- and high-voltage detect circuits for these and other power supplies are managed by the PMC, which provides enable and status bits for these voltage detection circuits.

The PMC sends a signal to the Reset Generation Module when power stabilization has been achieved. The Reset Generation Module then advances to the PHASE0 state (see Figure 8-1). During the power-up phase, the PMC does the following:

- Drives to a logic 0 all LVDxx (low-voltage detect) and HVDxx (high-voltage detect) signals monitored by the Power Management Control (PMC) module.
- Monitors its own power supply voltage (VDD_HV_PMC) to determine that the applied voltage is within a specified range.

- Monitors the core power supply voltage (VDD_LV_CORE) to determine that the applied voltage is within a specified range.
- Monitors the low- and high-voltage detect circuit power supply voltage to determine that the applied voltage is within a specified range.
- Holds the outputs of the internal low- and high-voltage detect circuits at a ground state until the PMC determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Enables all low- and high-voltage detect circuits once all power supplies are at their functional levels.
- Begins to monitor the various power supplies using the high- and low-voltage detect circuits.
- Provides status information to the Reset Generation Module (MC_RGM) indicating whether or not power is properly applied.

To exit power-up and enter PHASE0:

- All enabled destructive resets must be processed.
- All power supplies must be at their functional voltages.
- The PMC must signal the MC_RGM that all power supplies are at the required levels.

8.3.3 *PHASE0* Phase: analog supply initial configuration

This phase is entered:

- On exit from power-up phase.
- On PORST pin falling edge detection except for the initial power-up sequence.
- Immediately from PHASE2[DEST], PHASE3[DEST], IDLE[DEST], PHASE1[FUNC], PHASE2[FUNC], PHASE3[FUNC] or IDLE[FUNC], on the occurrence of a PORST falling edge event other than power-on reset.

During PHASE0:

- All digital modules are reset, including safety, security, and test modules.
- All trimming bits for analog modules are reset.
- Startup of the internal analog modules (PMC, IRCOSC, Flash memory, and I/Os) begins:
  - PMC determines that proper voltages are applied.
  - Flash memory performs internal self test and initialization.
  - I/O pins are ensured correctly configured: outputs are driven to known levels, inputs ignored.

To exit PHASE0 and enter PHASE1[DEST]:

- All enabled destructive resets must be processed.
- Gating PHASE0 must be released. See Table 10-4
- All processes initiated in ***PHASE0*** must be completed. This generally includes startup of the internal analog modules: PMC, IRCOSC, flash memory and I/Os.

This phase is typically completed in less than 20 µs. The chip is guaranteed to remain in this phase for at least 0.5 µs.

Note

The core voltage must rise above LVD112 upper threshold to exit PHASE0. This ensures significant hysteresis on low voltage supply avoiding spurious low voltage detection during the power-up process.

8.3.4 *PHASE1[DEST]* Phase: temporization and monitoring setup

This phase is entered on exit from PHASE0.

During PHASE1[DEST]:

- Initial configuration of the watchdog to monitor the reset sequence
- Temporization for reset signal propagation in case of reset

This phase is completed in less than 1 µs and corresponds to eight IRCOSC clock cycles.

Test modules are available during PHASE1[DEST], but their functionality is limited by the security module (no security waiver granted at this phase).

All PHASE1[DEST] tasks must be completed before entering PHASE2[DEST].

8.3.5 PHASE2[DEST] Phase: flash initial configuration

This phase is entered on exit from PHASE1[DEST].

During PHASE2[DEST]:

- Reset is released to the flash memory module by the Reset Generation Module.
- The flash memory module starts its initial configuration process.
- The flash memory initialization is performed by a state machine internal to the flash module.
- The system watchdog starts monitoring flash memory configuration execution so that a reset occurs in the event of configuration failure.

Note

System Watchdog is configured to generate destructive reset in case of flash memory failure during boot process. If, on boot completion, the watchdog event is converted to generate an FCCU fault only, application software should re-configure the MC_RGM watch-dog register bit to prevent destructive reset from being triggered.

All PHASE2[DEST] tasks must be completed before entering PHASE3[DEST].

This phase is completed in less than 200 µs.

8.3.6 PHASE3[DEST] Phase: device configuration

This phase is entered on exit from PHASE2[DEST].

During PHASE3[DEST], the System Status and Configuration Module (SSCM) starts by retrieving the Device Configuration Format (DCF) record from the TEST and UTEST flash memory areas, and uses the configuration and initialization information therein to:

- Configure the Self-Test Control Unit (STCU2) and the initial memory map of the device.

- Transfer reset vectors for the various cores to the Mode Entry Module (MC_ME). This information enables tests that the STCU runs and also provides the memory map setup information so that the STCU can test memory.
- Write trim values for analog modules including the Analog-to-Digital Converter, low- and high-voltage detect circuits, and the temperature sensor to their respective registers.

The DCF record programmed into the TEST flash memory area is generally written by the manufacturer. The portion of the DCF record programmed into the UTEST flash memory is usually written by the customer, although there is a default DCF record programmed into both the TEST and the UTEST flash memory areas during device production testing. In general, you write your own DCF record for the UTEST area to replace the default DCF.

The DCF record also provides configuration information for the following:

- Trimming of the analog modules (for example, voltage regulator, voltage detectors, I/Os, and oscillator)
    - Trim values are in DCF records in the TEST flash memory area.
    - Trim values are determined during production test and programmed into the TEST flash memory area.
- Device life cycle information:
    - Censorship configuration.
    - Debug security configuration.
    - Functional security configuration.
- Application configuration bits:
    - Watchdog configuration.
    - Safety execution directives (self test, LBIST/MBIST).
    - Boot Assist Flash options.
    - Oscillator startup during reset sequence configuration.

To exit PHASE3[DEST]: All processes that need to be completed in PHASE3[DEST] must be finished.

This phase depends on actual trim values, especially those used for safety execution directives. This device configuration is maintained until the next internal power-on reset. Each time a destructive reset event occurs, the SSCM takes values from the DCF record and writes them to the applicable registers.

During this phase, the internal STCU watchdog is started and is only cleared after completion of STCU configuration.

If self-test execution is the configuration information from the DCF record requests , the device moves to IDLE[DEST] and the STCU2 begins self-test. At the end of the test, the STCU2 asserts a functional interrupt causing the Reset Generation Module to move to PHASE1[FUNC]. If self-test execution is not requested, the device moves directly to IDLE[FUNC]. The Reset Generation Module determines whether self-test execution is enabled or not by examining an entry in the DCF record stored in the UTEST flash memory.

Note

When self-test execution is not requested, it is possible to maintain the device in PHASE3[DEST] by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.

8.3.7 IDLE[DEST] Phase: self-test execution

This phase is entered upon exit from PHASE3[DEST] if the execution of self test is enabled.

During the IDLE[DEST] phase:

- Dedicated watchdog is configured to monitor self-test execution completion within expected time. The length of this phase is variable depending on the self-test requirements.
- The Self-Test Control Unit (STCU2) executes all tests specified by the DCF information retrieved in PHASE3[DEST].
- The self-test engine updates the self-test completion flag and triggers an associated functional reset on completion of self test.

The length of this phase is variable depending on the self-test requirements.

Note

When the SSCM is decoding the Device Configuration field, it sets and clears various flags in STCU registers and memory that can later be examined by the SSCM to determine which self-test directives must be run.

Exiting IDLE[DEST] and moving to PHASE1[FUNC]:

- The IDLE[DEST] phase is automatically exited when the self-test code generates a functional reset on completion.
- The functional reset causes the reset generation module to move to the PHASE1[FUNC] phase.

Note

In case of unrecoverable fault detection during STCU execution, unrecoverable fault reset is triggered and the chip re-enters PHASE1[DEST] .

8.3.8 PHASE1[FUNC] Phase: temporization and monitoring setup

This phase is entered either:

- On exit from IDLE[DEST]
- Immediately from PHASE2[FUNC], PHASE3[FUNC], or on IDLE[FUNC] when a non-masked external or "functional" reset event occurs, providing the source of the reset event has not been configured to trigger a short reset sequence

During PHASE1[FUNC]:

- Initial configuration of the watchdog to monitor reset process
- Temporization for reset signal propagation in case of reset

To exit PHASE1[FUNC] and enter PHASE2 [FUNC]:

- All enabled, non-shortened functional resets must be processed.
- All processes started in PHASE1[FUNC] must be completed.
    - Initial configuration of the watchdog to monitor reset process
    - Temporization for reset signal propagation in case of reset

This phase is completed in ~1 µs or less. During this phase, test modules are available.

8.3.9 PHASE2[FUNC] Phase: flash initial configuration

This phase is entered on exit from PHASE1[FUNC].

During the PHASE2[FUNC] phase:

- The reset signal is released to the flash memory.
- Flash initialization and configuration begins using a state machine that is internal to the flash memory module.
- A watchdog monitors flash initialization to ensure a reset in case of flash memory configuration failure.

The reset state machine exits PHASE2[FUNC] and enters PHASE3[FUNC] on verification that all processes started in PHASE2[FUNC] are completed.

This phase is completed in ~200 µs .

8.3.10 PHASE3[FUNC] Phase: device configuration monitoring

PHASE3[FUNC] is entered:

- On exit from PHASE2[FUNC]
- Immediately from IDLE[FUNC] when an enabled, short functional reset event occurs.

During PHASE3[FUNC], configuration information contained within the UTEST flash memory sector is checked against information extracted during PHASE3[DEST] and written to the Self-Test Control Unit (STCU2). In case of mismatches, a dedicated boot fault is triggered within the FCCU which, if enabled, can cause a system reset. The purpose of PHASE3[FUNC] is to verify that the DCF information transferred to the STCU, MC_ME and other modules, in PHASE3[DEST], was performed correctly and has not been corrupted.

The Reset Generation Module exits PHASE3[FUNC] and enters the IDLE [FUNC] on verification of the following:

- All processes started in PHASE3[FUNC] are completed.

- PORST and ESR0 pin are not forced low externally.
- A minimum number of cycles have elapsed since the last enabled, shortened functional reset event.

After all PHASE3[FUNC] internal actions have been completed, it is possible to maintain the device within PHASE3[FUNC] by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes in order to connect through the JTAG port to the internal debugger. In the case of a secured device, it is possible to submit the debug password to access device and/or HSM debug through the JTAG module. The HSM responds by performing certification of the provided password and eventually may enable debug access to the different cores.

8.3.11 IDLE[FUNC] Phase

This is the Reset Generation Module's final phase of the reset sequence. It is entered either on exit from PHASE3[DEST] (if self test was not enabled) or at the completion of PHASE3[FUNC]. When the IDLE[FUNC] phase is reached, the MC_RGM releases control of the system to the System Status and Control Module. The SSCM, which started in PHASE3[FUNC], runs and waits for a signal from the MC_RGM indicating that the MC_RGM is now in the IDLE state. The MC_RGM then waits for new reset events that can trigger a reset sequence. The SSCM continues with the system boot-up sequence.

8.4 MC_RGM passes boot sequence control to the System Status and Control Module

The SSCM continues the boot-up sequence after the Self-Test Control Unit (STCU2) finishes executing the MBIST and LBIST functions that were initiated by the MC_RGM during the IDLE[DEST] state.

A flow chart of the boot-up sequence is shown in Figure 8-2, Figure 8-3, Figure 8-4, Figure 8-5, and Figure 8-6.

8.4.1 Reset sequence flow based on initial device condition

The boot-up sequence takes different paths depending on which modules are enabled and which header files are present. For instance, a device shipped from the factory has pre-installed programs in the Boot Assist Flash and the Boot Assist ROM. Also DCF records are programmed into the TEST flash memory area and the UTEST flash memory area.

- Boot Assist Flash contains factory code which cannot be modified by user.
- Boot Assist ROM contains factory code which cannot be modified by user.
- TEST DCF record cannot be modified by user.
- UTEST flash memory contains a default DCF record which can be replaced with a user file.

After the chip's flash memory has been programmed, it contains the CPU boot header file, and, if required, an HSM boot header file, together with the user application. This boot header is used by the boot-up sequence flow managed by the SSCM module.

8.4.2 Possible boot-up sequences

This section describes possible boot-up sequences.

8.4.2.1 Enter JTAG test mode

- Used for JTAG testing (non-user mode)
- Interface to JTAG port Development Equipment for Code Development

8.4.2.2 Enable IOP and Boot CPU (CPU_0 or CPU_1)

- Normal operation for I/O Processor (IOP)
- Normal operation for the selected boot CPU

8.4.2.3 Enable IOP, HSM and enable the Boot CPU (CPU_0 or CPU_1)

- Normal operation for I/O Processor (IOP)
- Normal operation for selected boot CPU
- Normal Hardware Security Module (HSM) operation

8.4.2.4 Enable IOP and/or HSM and enter serial boot mode if no application code is present

- Mode used to download program code and place it in RAM
- Mode used to start downloaded application program

The following sections detail the exact flow through the reset sequence to achieve the various outcomes.

8.4.3 Sequence from MC_RGM IDLE to IOP, HSM, and boot CPU running code

Here the operation of the IOP and HSM is desired. A boot CPU is selected and starts executing application code; all of the processor cores are also started: CPU_0, CPU_1, Checker Core_0s, HSM, and IOP.

8.4.3.1 Initial conditions

- Boot Assist Flash bypass mode is NOT enabled (execute BAF code).
- HSM is enabled.
- Application code for the IOP is programmed into Boot Assist Flash memory.
- Application code for the HSM is programmed into flash memory.
- Application code for the boot CPU is programmed into flash memory.
- A valid boot header for CPU_0, CPU_1, and the Checker Core_0s has been programmed into flash memory with:
  - Boot Header_ID
  - Boot CPU selection
  - Boot CPU start address (address of first instruction in application code)
  - CPU_0 reset vector
  - CPU_1 reset vector
  - Checker Core_0s reset vector
- A valid boot header for HSM has been written into flash memory with:

- HSM_Boot_Header_ID
- Start address of HSM application code

- The Reset Generation Module (MC_RGM) has passed control to the SSCM during the IDLE[FUNC] state.

8.4.3.2 Reset Generation Module has entered idle state

The MC_RGM enters the IDLE[FUNC] state and completes all the tasks in defined the IDLE[FUNC] state. It then signals the SSCM that it has become idle and the SSCM can take control of boot-up.

8.4.3.3 LBIST and MBIST start execution (state 1 and 2)

The MC_RGM enters the IDLE[FUNC] state and completes all tasks. It then signals the SSCM that it has become idle and the SSCM can take control of the boot-up sequence.

At this point:

- LBIST and MBIST have completed their test cycles.
- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.3.4 Is a JTAG request pending? (state A)

The SSCM now controls the boot-up sequence and begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems.

The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.3.5 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM determines whether or not the Boot Assist Flash bypass mode is enabled. If enabled, any eventual code in the BAF is not executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the BAF bypass mode. With the BAF bypass mode disabled, code in the Boot Assist Flash is always executed during a boot-up sequence.

The factory BAF code also runs the Serial Boot Loader, setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins IOP (I/O Processor) execution at the start address specified in the file header.

8.4.3.6 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the Hardware Security Module (HSM) is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is enabled.

8.4.3.7 Locate HSM Boot Header (state 44)

The SSCM is responsible for locating the HSM boot header. The SSCM reads several memory locations in flash memory to determine whether a valid HSM boot header is present. Specifically, the SSCM reads locations 0x60_C000, 0x61_0000 and 0x62_0000. A valid HSM boot header consists of three words: an HSM_Boot_Header_ID, a start address for the HSM application code, and a third word that is reserved for future use. By definition a valid HSM_Boot_Header_ID is 0xFFFF_0000_FFFF_0000.

If a valid HSM_Boot_Header_ID is found, the SSCM transfers the start address to the Mode Entry Module (MC_ME). When reset is released to the HSM, the HSM start address is transferred from the MC_ME to the HSM and HSM code execution begins at that point.

8.4.3.8 Valid Hardware Security Module header file? (state D)

In the present scenario, a valid HSM boot header is present as determined in state 44.

8.4.3.9 Provide reset vectors to IOP and HSM (state 25—go to state 22 and 26)

Under the control of the SSCM, the HSM start address obtained from the HSM boot header is transferred to the MC_ME.

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code is transferred to the MC_ME. The SSCM releases reset to the IOP and HSM and then turns on the clock to both modules. The BAF code start address is transferred from the MC_ME to the IOP. At this time, the HSM code start address is also transferred from the MC_ME to the HSM under the control of the SSCM.

8.4.3.10 HSM code and BAF code execution begins (state 26)

After the reset signal is released and the respective clock is turned on to the HSM and the IOP, code execution begins. The HSM begins executing its application code and the IOP begins execution of code in the Boot Assist Flash.

8.4.3.11 Verification of HSM host boot block (state 27)

HSM applications code can optionally be written to confirm that the correct start address for the HSM applications code was obtained from the HSM boot header file. This step is optional and requires that the HSM applications code and the HSM boot header file match.

If a new revision of the HSM code is created with a start address that is different from the prior revision, the HSM applications code must be modified to take into account that there is a different start address. The HSM Boot Header must also be appropriately modified.

8.4.3.12 Set HSM Boot Ready flag (state 28)

The HSM hardware sets the HSM Boot Ready flag when the HSM becomes fully initialized.

8.4.3.13 Execution of HSM application code begins (state 29)

The HSM has been properly enabled, a valid start address for code execution has been supplied and the HSM begins normal code execution.

8.4.3.14 IOP begins BAF code execution (state 22)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

8.4.3.15 IOP applies device settings (state 23)

The IO Processor, executing code in the Boot Assist Flash, configures the MPC5777M with various initialization parameters. The System Status and Configuration Module (SSCM) has already applied some configuration information from the DCF record.

8.4.3.16 Calibration Sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production MPC5777M (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The emulation and debug device (ED) consists of a regular production MPC5777M (PO) and a special Buddy Device (BD) in a single package.

The I/O Processor (IOP) executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is detected, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of MPC5777M reset.

In this example, it is assumed that an emulation and debug device (ED) is not being used in place of a regular production MPC5777M.

8.4.3.17 Is HSM enabled? (decision point E)

The IOP, executing BAF code, checks to see if the HSM is enabled. In this example, it is assumed that the HSM is enabled.

8.4.3.18 Wait for HSM to set HSM_Boot_Ready flag? (decision point F)

The IOP determines whether it must wait until the HSM becomes ready before the IOP begins a search for the boot header file. If the IOP is directed to wait for the HSM to set its HSM_Boot_Ready flag, the IOP goes to state 30 until the "HSM_Boot_Ready" flag is set. Afterwards, the IOP begins a search for the Boot Header file in state 31.

If the IOP is not directed to wait for the HSM to set its HSM_Boot_Ready flag, the IOP immediately begins a search for the boot header file in state 31.

8.4.3.19 Wait until HSM_Boot_Ready flag sets (state 30)

The IOP waits until the HSM_Boot_Ready flag is set.

8.4.3.20 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. This is the header file for the CPU that starts executing the your application code. Generally, the BOOT CPU is either Core_0 or Core_1.

The Boot CPU header is programmed into flash memory at the same time your application program is programmed into flash memory. The IOP searches for the boot header file which can be located at one of eight addresses:

Note

Addresses are searched in the following order.

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid boot header begins with the value 0x005A.

The boot header structure consists of seven words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector

- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

- Bit 0 selects the I/O Processor
- Bit 1 selects CPU_0
- Bit 2 selects CPU_1
- Bit 3 selects the Checker Core_0s
- Bits 4–15 are not implemented

Note

Checker Core_0s can only be used if CPU_0 is also used. Both cores must have the same reset vector.

8.4.3.21 Valid boot header is found (state N)

In the present example, a valid boot header is assumed to be present in flash memory.

8.4.3.22 Decode boot CPU (state 32)

The IOP decodes the boot header and then writes the reset vectors for CPU_0, CPU_1, and the Checker Core_0s to the Mode Entry Module.

8.4.3.23 Lock BAF access (state 33)

Further access to the BAF is locked against further code execution, but it is readable.

8.4.3.24 Boot From Internal flash (state 34)

The MC_ME transfers the reset vectors to the respective CPUs and code execution for each of the cores begins at their respective reset vector address.

8.4.4 Sequence from MC_RGM IDLE to IOP and Boot CPU running code (HSM disabled)

Here the IOP is desired, but not the HSM. A Boot CPU is selected and starts executing application code; it also starts all of the processor cores except HSM: CPU_0, CPU_1, Checker Core_0s, and IOP.

8.4.4.1 Assumptions

- Boot Assist Flash bypass mode is not enabled (execute BAF code).
- HSM is not enabled.
- Application code for the IOP is programmed into Boot Assist Flash memory.
- Application code for the Boot CPU is programmed into flash memory.
- A valid boot header for CPU_0, CPU_1 and Checker Core_0s has been programmed into flash memory.
    - Boot Header_ID
    - Boot CPU selection
    - Boot CPU start address (address of first instruction in application code)
    - CPU_0 reset vector
    - CPU_1 reset vector
    - Checker Core_0s reset vector
- The MC_RGM has passed control to the SSCM during the IDLE[FUNC] state.

8.4.4.2 Reset Generation Module has entered idle state (state 1 and 2)

The Reset Generation Module (MC_RGM) enters the IDLE[FUNC] state and completes all tasks. The MC_RGM then signals the System Status and Control Module (SSCM) that it has become idle and the SSCM can take control of the boot-up sequence.

At this point:

- LBIST and MBIST have completed their test cycles.
- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.4.3 Is a JTAG request pending? (state A)

The SSCM is now in charge of the boot-up sequence. It begins by determining whether a JTAG request is pending. The JTAG test circuitry is for production test and used by software development systems.

The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.4.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether the Boot Assist Flash bypass mode is enabled. If this mode is enabled, any eventual code in the BAF is not executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. This means that the code in the Boot Assist Flash is always executed during a boot-up sequence.

The factory BAF code also runs the serial boot loader, setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins IOP execution at the start address specified in the file header.

8.4.4.5 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the HSM is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is not enabled.

8.4.4.6 Provide reset vectors to IOP (state 21)

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code is transferred to the MC_ME.

The SSCM releases reset to the I/O Processor (IOP) and then turns on the clock to the IOP. The BAF code start address is transferred to the IOP at this time under the control of the SSCM. Execution of the Boot Assist Flash code begins.

8.4.4.7 BAF code execution begins (state 22)

After the reset signal is released to the IOP, BAF code execution begins.

8.4.4.8 The IOP executes code to apply device setting (state 23)

The IOP executes BAF code, written by the manufacturer, to apply specific device settings to the MPC5777M.

The BAF code begins by initializing registers and applying device settings. The BAF code, written by the device manufacturer, changes from device to device.

8.4.4.9 Calibration sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production MPC5777M (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The ED consists of a PO and a special Buddy Device (BD) in a single package.

The IOP executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is present, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of MPC5777M reset.

In this example, it is assumed that an ED is not being used in place of a regular MPC5777M production device.

8.4.4.10 Is HSM enabled? (decision point E)

The IOP determines whether the HSM is enabled by examining a status bit in the HSM. In the present example, it is assumed that the HSM is not enabled.

8.4.4.11 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. In the present example, it is assumed that a valid Boot Header is present.

This is the header file that specifies which Boot CPU starts executing the user application code. Generally, the Boot CPU is Core_0 or Core_1. The boot header file also contains the reset vector addresses for Core_0, Core_1, and the Checker Core_0s.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The IOP searches for the boot header, which can be located at one of eight addresses:

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of the following words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

- Bit ‘0’ selects the I/O Processor
- Bit ‘1’ selects CPU_0,
- Bit ‘2’ selects CPU_1,
- Bit ‘3’ selects the Checker Core_0s.
- Bits ‘4–15’ are not implemented.

8.4.4.12 Is a valid boot header present? (decision point N)

In the present example, a valid boot header is assumed to be present in flash memory.

8.4.4.13 Decode boot CPU (state 32)

The I/O Processor decodes the boot header and then writes the reset vectors for CPU_0, CPU_1, and the Checker Core_0s to the Mode Entry Module (MC_ME).

8.4.4.14 Lock BAF access (state 33)

Further access to the Boot Assist Flash is locked.

8.4.4.15 Boot from internal flash (state 34)

The MC_ME transfers the reset vectors to the respective CPUs and code execution for each of the cores begins at their respective reset vector.

8.4.5 Path from MC_RGM IDLE to serial boot mode

Here no application program or valid boot headers for the HSM or CPU_0, CPU_1, or Checker Core_0s is programmed into flash memory. It is an unusual scenario, but it is shown here for illustrative purposes.

8.4.5.1 Initial conditions

- No valid boot header for CPU_0, CPU_1, and Checker Core_0s has been programmed into flash memory.
- BAF bypass mode is NOT enabled (execute BAF code).
- The HSM is enabled.
- Application code for the IOP is programmed into the BAF.
- Application code for the HSM is not programmed into secure flash memory.
- Application code for the Boot CPU is not programmed into flash memory.

- A valid boot header for the HSM has not been written into flash memory.
- The Life Cycle variable is set to FACTORY, indicating an unprogrammed device.
- The MC_RGM has passed control to the SSCM during the IDLE[FUNC] state.

8.4.5.2 Reset Generation Module has entered idle state (state 1 and 2)

The MC_RGM enters the IDLE[FUNC] state and completes all tasks. The MC_RGM then signals the SSCM that it has become idle and the SSCM can take control of the boot-up sequence.

At this point:

- LBIST and MBIST have completed their test cycles.
- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.5.3 Is a JTAG request pending? (state A)

The SSCM now controls the boot-up sequence and begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems.

The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.5.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether the BAF bypass mode is enabled. If enabled, any code in the BAF is not executed. The BAF bypass mode is enabled/ disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. This means that the code in the BAF is always executed during a boot-up sequence.

The BAF code written by the device manufacturer also runs the serial boot loader. This involves setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins the IOP execution at the start address specified in the header portion of the program file.

8.4.5.5 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the HSM is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is enabled.

8.4.5.6 Locate HSM boot header (state 44)

The SSCM is responsible for locating the HSM boot header. The SSCM reads several memory locations in flash memory to determine whether a valid HSM boot header is present. Specifically, the SSCM reads locations 0x60_C000, 0x61_0000 and 0x62_0000. A valid HSM boot header consists of three words: an HSM_Boot_Header_ID, a Start Address for the HSM application code, and a third word that is reserved for future use. By definition, a valid HSM_Boot_Header_ID is 0xFFFF_0000_FFFF_0000.

8.4.5.7 Valid Hardware Security Module header file? (state D)

In the present scenario, no valid HSM boot header is present. In this case, the System Status and Control Module takes the same path as if the HSM was not enabled at decision point C.

8.4.5.8 Provide reset vectors to IOP (state 21)

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code in the Boot Assist Flash is transferred from the DCF record to the MC_ME by the SSCM.

The SSCM releases reset to the IOP and then turns on the clock to the IOP. The clock to the HSM is turned off in this state even though the HSM was originally enabled. Because no valid header file for the HSM is found, the IOP goes to state 21 from decision point D, which turns the HSM clock off. The start address is transferred to the IOP at this time under the control of the SSCM and the IOP begins execution of the BAF code.

8.4.5.9 IOP begins BAF code execution (state 22)

After the reset signal is released and the respective clocks are turned on to the IOP, the IOP begins execution of code in the Boot Assist Flash.

8.4.5.10 IOP continues BAF code execution (state 23)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

8.4.5.11 Calibration Sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production MPC5777M (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The ED consists of a PO and a special Buddy Device (BD) in a single package.

The IOP executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is present, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of MPC5777M Reset.

In this example, it is assumed that an ED is not being used in place of a regular MPC5777M production device.

8.4.5.12 IOP checks to determine if HSM is enabled (state E)

The IOP, executing BAF code, checks to see if the HSM is enabled. In the present case, the SSCM has already disabled the clock to the HSM. Therefore, the HSM is not enabled.

8.4.5.13 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. In the present example, it is assumed that no boot header file is present.

This is the header file for the CPU that starts executing the user application code. Generally, the BOOT CPU is Core_0 or Core_1.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The IOP searches for the boot header, which can be located at one of eight addresses:

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of the following words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

- Bit ‘0’ selects the I/O Processor
- Bit ‘1’ selects CPU_0,
- Bit ‘2’ selects CPU_1,
- Bit ‘3’ selects the Checker Core_0s.
- Bits ‘4–15’ are not implemented.

8.4.5.14 Valid boot header is not found (decision point N)

The IOP, executing code in the BAF, does not find a valid boot header file in flash memory. Because there is no valid boot header, serial boot mode must be entered to obtain an application program.

Note

This mode uses a defined protocol to receive a program over a serial port via the LINFlexD port. The IOP then jumps to the start address of the program and begins execution.

8.4.5.15 Initialize RAM (state 35)

The IOP, executing code in the Boot Assist Flash, initializes RAM in preparation for entering serial boot mode. Serial boot mode configures the LINFlexD port to receive code from an external source. The LINFlexD port comes in a number of hardware configurations. The chip uses LINFlexD, which implements a basic UART. Code in the BAF operates the LINFlexD module and implements the protocol.

8.4.5.16 Enter serial boot mode (state 14)

Figure 8-6 shows the boot loader flow.

8.4.5.17 Life Cycle check indicates FACTORY (decision point G)

There are two possible settings for the Life Cycle parameter:

- FIELD indicates that application code has been programmed into the flash memory of the device. Serial boot mode is exited and IOP enters a ‘wait’ mode until a watchdog reset occurs.
- FACTORY means that there is no application code programmed into the flash memory. For the present example a setting of FACTORY is assumed.

8.4.5.18 Receive and install application code

In general, a serial link is implemented using the LINFlexD port and a particular protocol for data transmission has been defined and implemented in the Boot Assist Flash code. Once the LINFlexD port is configured, the serial boot loader waits to receive the code.

Note

There is a software watchdog timer that is started by the BAF code. If a program is not received before the watchdog timer timeout, a reset occurs.

- Configure LINFlexD module and external pins (state 37)
- Wait for START WORD (state 38)
- Receive START WORD (state 39)
- Receive START ADDRESS (state 40)
- Receive DOWNLOAD SIZE (state 41)
- Receive CODE (state 42)
- Branch to START ADDRESS (state 43)

Application code, using the IOP, begins at the received START ADDRESS.

8.4.6 Path from MC_RGM IDLE with IOP and HSM enabled to watchdog timer timeout

Here the operation of the IOP and the HSM is desired but there is no valid boot header file or HSM header file and the Life Cycle variable is set to FIELD, indicating that the device was programmed with application code and the appropriate boot header files.

This path is only taken if a catastrophic event has occurred or the device is improperly programmed.

8.4.6.1 Initial conditions

- BAF bypass mode is not enabled (execute BAF code).
- The HSM is enabled.
- A valid boot header for the HSM has not been programmed into flash memory.
- A valid boot header for CPU_0, CPU_1 and Checker Core_0s has not been programmed into flash memory.
  - Boot Header_ID
  - Boot CPU selection

- Boot CPU start address (address of first instruction in application code)
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

- The Life Cycle variable is set to FIELD, indicating a programmed device.
- The MC_RGM has passed control to the SSCM during the IDLE[FUNC] state.

8.4.6.2 Reset Generation Module has entered idle state (state 1 and 2)

The MC_RGM enters the IDLE[FUNC] state and completes all tasks. It then signals the SSCM that the MC_RGM has become idle and the SSCM can take control of the boot-up sequence.

At this point:

- LBIST and MBIST have completed their test cycles.
- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

8.4.6.3 Is a JTAG request pending? (state A)

The SSCM now controls the boot-up sequence and begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems.

The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

8.4.6.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether or not the boot assist flash bypass mode is enabled. If this mode is enabled, any eventual code in the Boot Assist Flash is not executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. This means that the code in the BAF is always executed during a boot-up sequence.

The factory BAF code also runs the serial boot loader, setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins IOP execution at the start address specified in the file header.

8.4.6.5 Is the Hardware Security Module enabled? (state C)

The SSCM examines a status bit to determine if the HSM is enabled. A DCF record in the TEST area is responsible for enabling/disabling the HSM. In the present example, it is assumed that the HSM is enabled.

8.4.6.6 Locate HSM boot header (state 44)

The SSCM is responsible for locating the HSM boot header. The SSCM reads several memory locations in flash memory to determine whether a valid HSM boot header is present. Specifically, the SSCM reads locations 0x60_C000, 0x61_0000 and 0x62_0000. A valid HSM boot header consists of three words: an HSM_Boot_Header_ID, a Start Address for the HSM application code, and a third word which is reserved for future use. By definition, a valid HSM_Boot_Header_ID is 0xFFFF_0000_FFFF_0000.

8.4.6.7 Valid Hardware Security Module header file? (state D)

In the present scenario, no valid HSM boot header is present. In this case, the SSCM takes the same path as if the HSM was not enabled at decision point C.

8.4.6.8 Provide reset vectors to IOP (state 21)

During PHASE3[DEST] of the Reset Generation Module's sequence, the start address for execution of the BAF code in the Boot Assist Flash is transferred from the DCF record to the MC_ME.

The SSCM releases reset to the IOP and then turns on the clock to the IOP. The clock to the HSM is turned off in this state even though the HSM was originally enabled. Because no valid header file for the HSM is found, the IOP goes to state 21 from decision point D, which turns the HSM clock off. The start address is transferred to the IOP at this time under the control of the SSCM and the IOP begins execution of the Boot Assist Flash code.

8.4.6.9 IOP begins BAF code execution (state 22)

After the reset signal is released and the respective clocks are turned on to the IOP, the IOP begins execution of code in the Boot Assist Flash.

8.4.6.10 IOP continues BAF code execution (state 23)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

8.4.6.11 Calibration sequence (state 24)

This state is used to detect the presence of an emulation and debug device (ED) normally used in place of a regular production MPC5777M (PO). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The ED consists of a PO and a special Buddy Device (BD) in a single package.

The IOP executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is present, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of MPC5777M reset.

In this example, it is assumed that an ED is not being used in place of a regular MPC5777M production device.

8.4.6.12 IOP checks to determine if HSM is enabled (state E)

The IOP, executing BAF code, checks to see if the HSM is enabled. In the present case, the SSCM has already disabled the clock to the HSM. Therefore, the HSM is not enabled.

8.4.6.13 IOP searches for boot header (state 31)

The IOP continues code execution by searching for the boot header. In the present example, it is assumed that no boot header file is present.

This is the header file for the CPU that starts executing the user application code. Generally, the BOOT CPU is Core_0 or Core_1.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The IOP searches for the boot header, which can be located at one of eight addresses:

- 0xFC_0000
- 0xFC_4000
- 0xFC_8000
- 0xFC_C000
- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of the following words:

- Boot Header_ID/Boot_CPU
- Start address
- Configuration bits
- Configuration bits
- CPU_0 reset vector
- CPU_1 reset vector
- Checker Core_0s reset vector

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

- Bit ‘0’ selects the I/O Processor
- Bit ‘1’ selects CPU_0,
- Bit ‘2’ selects CPU_1,
- Bit ‘3’ selects the Checker Core_0s.
- Bits ‘4–15’ are not implemented.

8.4.6.14 Valid boot header is not found (decision point N)

The IOP, executing code in the BAF, enters serial boot mode to obtain an application program as it does not find a valid boot header file in flash memory.

Note

This mode uses a defined protocol to receive a program over a serial port via the LINFlexD port. The IOP then jumps to the start address of the program and begins execution.

8.4.6.15 Initialize RAM (state 35)

The IOP, executing code in the BAF, initializes RAM areas in preparation for serial boot mode. This mode configures the LINFlexD port to receive code from an external source using a basic UART protocol.

8.4.6.16 Enter serial boot mode (state 14)

Figure 8-6 shows the boot loader flow.

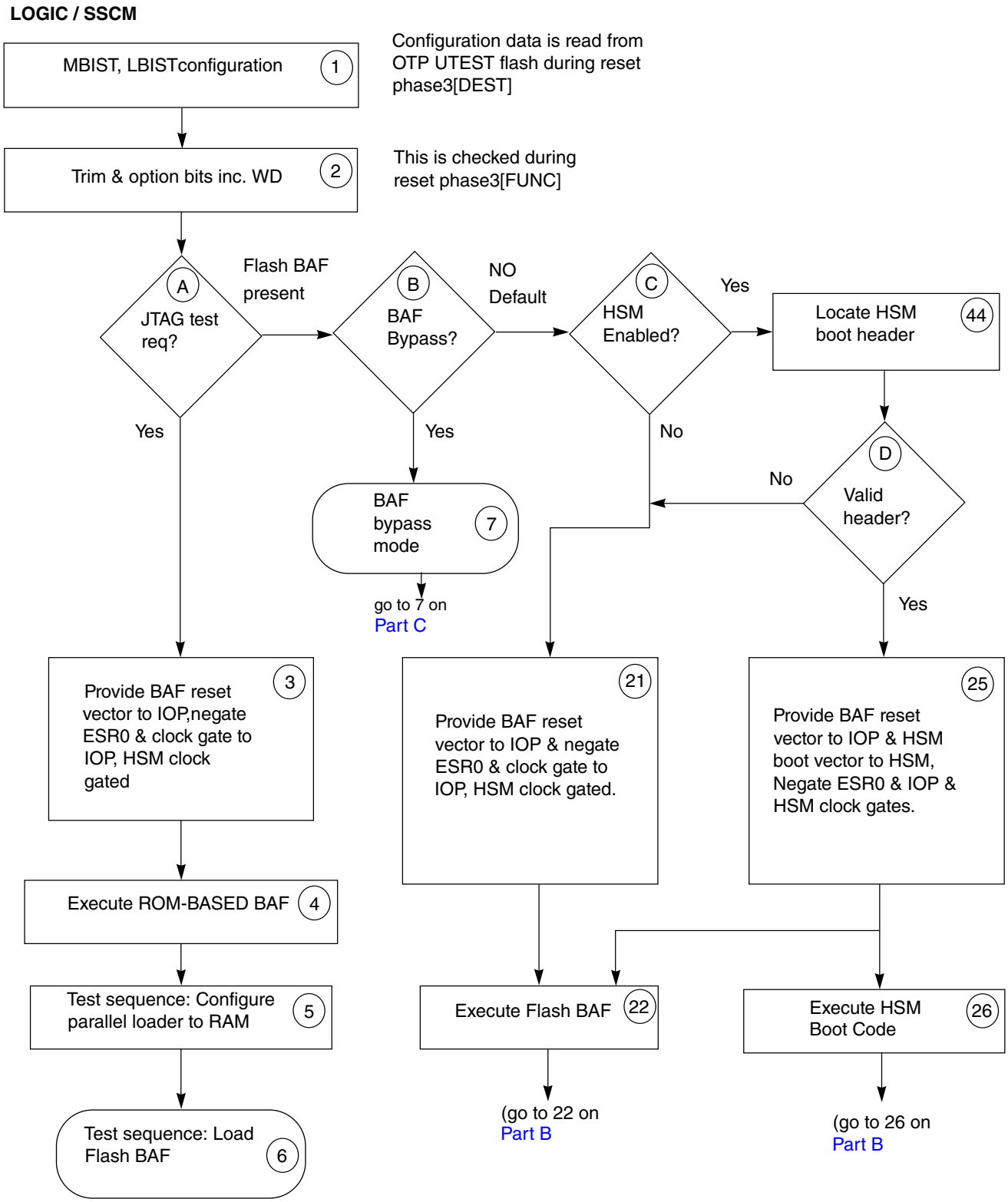
8.4.6.17 Life Cycle check indicates FIELD (decision point G)

There are two settings for the Life Cycle parameter: FIELD and FACTORY.

FIELD indicates that application code has been programmed into the flash memory. In this case, the IOP enters a ‘wait’ mode until a watchdog reset occurs.

8.4.6.18 Stop—wait for watchdog (state 36)

This state is entered when the chip’s program flash memory is incorrectly programmed.



Boot Flow and interaction with other modules (part 1)

Figure 8-2. Boot-up sequence part A

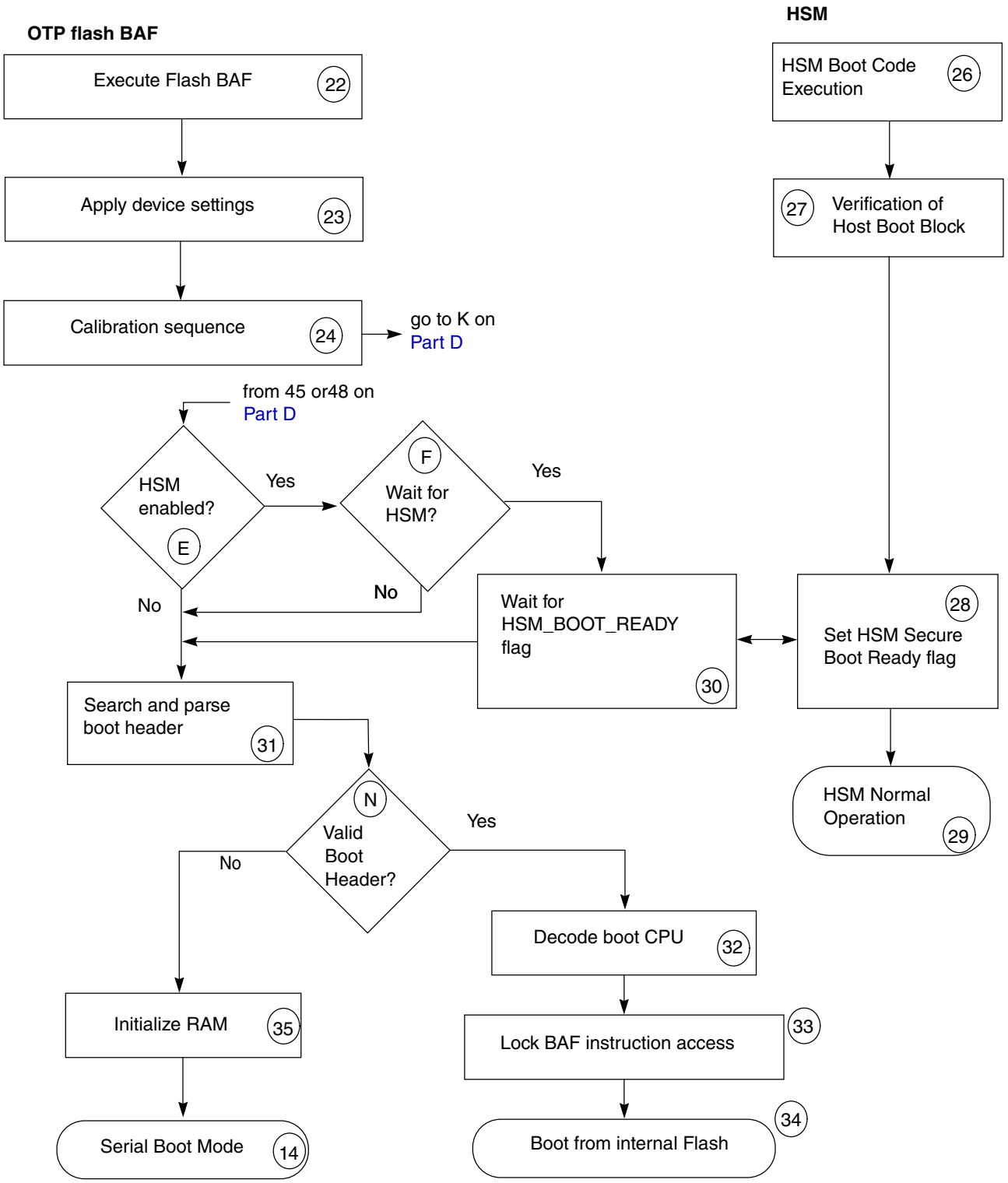


Figure 8-3. Boot-up sequence part B

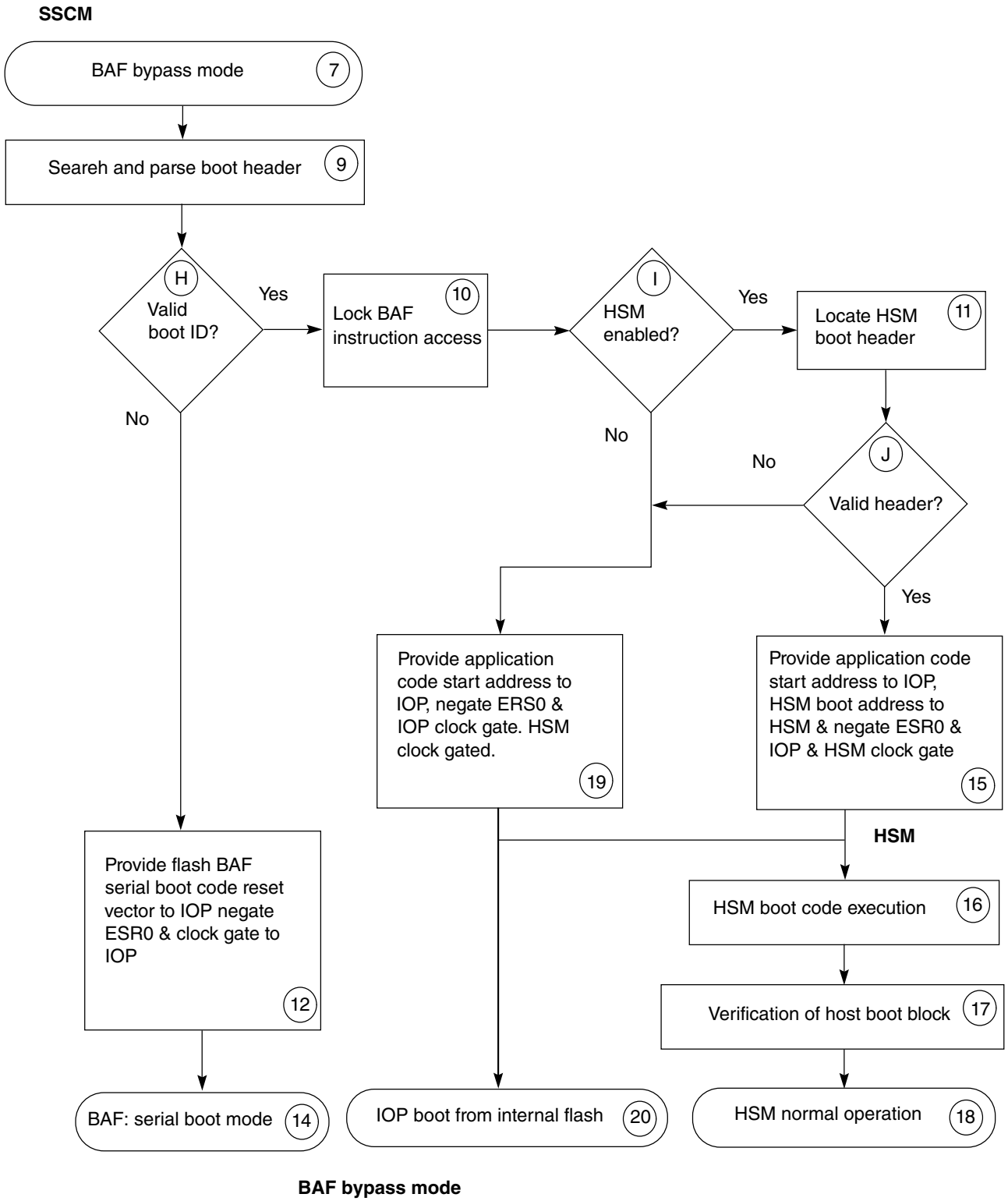
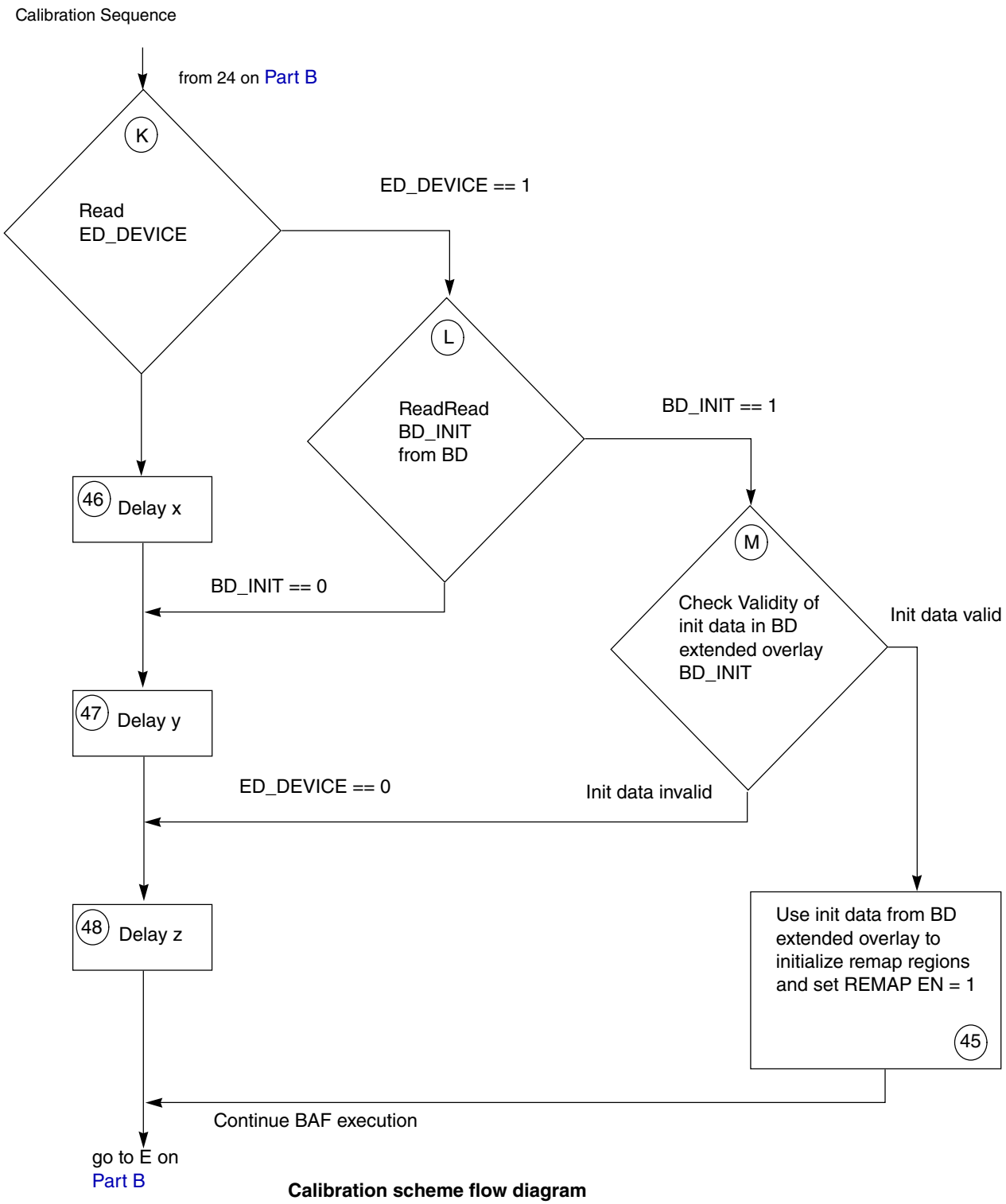
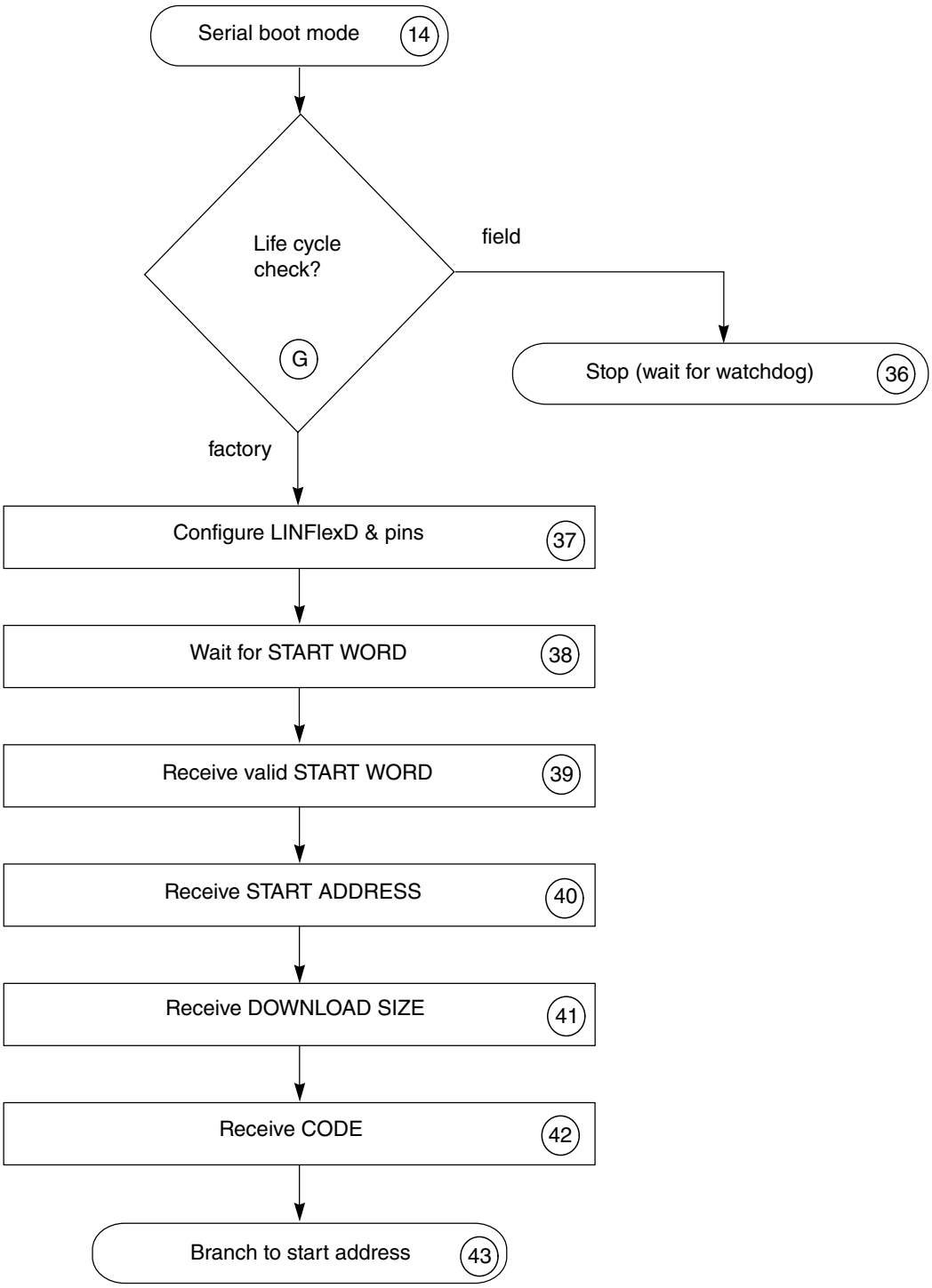


Figure 8-4. Boot-up sequence, part C



Calibration scheme flow diagram

Figure 8-5. Boot-up sequence, part D



Serial boot process flow diagram

Figure 8-6. Boot-up sequence, part E

Chapter 9
Device Configuration Format (DCF) Records

9.1 Introduction

Device Configuration Format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record consists of a pointer to the location of a register internal to the device and the respective data.

There are two broad categories of DCF records, TEST DCF Records and UTEST DCF Records:

- TEST DCF records are developed by the factory. They can be used to write any DCF client, but are used mainly to:
  - Program registers involved in trimming trip points for voltage comparators
  - Adjusting analog to digital voltage supplies
  - Trim oscillator frequencies
  - Enable RAM repair

NOTE

The TEST DCF Records are programmed into TEST flash during production and cannot be modified. TEST flash is not visible to the user.

- UTEST DCF records can be either:
  - Factory programmed during production testing (see “UTEST flash memory map” table in Flash memory maps).
  - User programmed when application code is written to the flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the factory UTEST DCF records. The user-defined DCF records set up the initial memory map, define which tests the Self-Test Control Unit (STCU2) runs during the boot sequence, enable the HSM, assign flash memory blocks to specific tamper detect regions, assign memory blocks as OTP and assign flash blocks to be associated with specific password groups.

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPUs and the DCF records must be properly programmed into their respective flash memories.

When power is applied to a properly programmed device:

1. The Power Management Controller (PMC) takes control of the device until the system power supplies have reached predefined levels.
2. The PMC then signals the Reset Generation Module (RGM) to begin the boot sequence.
3. During boot, the RGM enables the System Status and Control Module (SSCM) to read the device configuration records and write information to specific registers.
4. After the reset process has completed, the STCU, depending on its DCF values, may start the self test.

9.2 DCF clients

DCF clients are 32-bit hardware registers inside a module that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF Clients have a default value before any DCF Records are written, and may have special writing constraints, such as 'Write Once' or only allowing bits to be written from '1' to '0' or vice versa. DCF clients need not implement all 32 bits. DCF clients may be designated 'TEST DCF record only', so that only DCF Records stored in TEST flash can write to the DCF Client.

A list of DCF clients is shown in Table 9-6. (TEST DCF Record Only Clients are not shown as these are not accessible by the user).

The DCF Records select the target DCF client via a 30-bit field in the DCF Record consisting of:

- A 15-bit Chip-Select field—each module that includes DCF clients is assigned a Chip Select during chip definition.
- A 15-bit Address field—the address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

9.3 DCF records

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST flash memory beginning at 0x0040_0300.

The structure of a DCF Record is shown in Table 9-1.

Table 9-1. DCF Record Structure

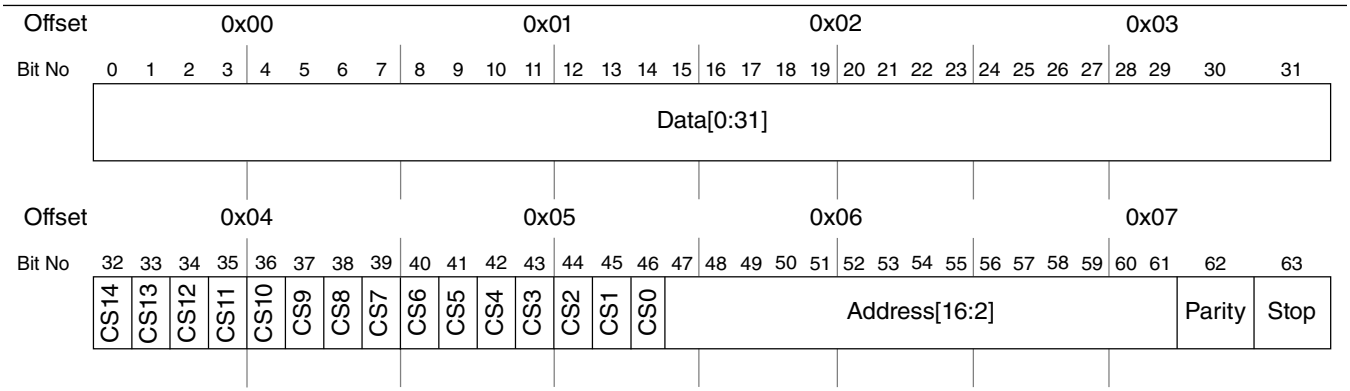


Figure 9-1. DCF Record Structure

Table 9-2. DCF Record Field Description

Field	Name	Description
0–31	Data[0:31]	32 bits of data that is to be written to the DCF Client
32–46	CS*n*	Chip Select *n*. **NOTE:** Only assert one chip per DCF record to select the target module for the DCF client. All other Chip Selects should be negated. 0 Chip Select is asserted 1 Chip Select is negated
47–61	Address[16:2]	Address of the DCF client within the selected module. **NOTE:** Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter
62	Parity	Parity Bit for the DCF Record. **NOTE:** This bit is NOT implemented for DCF Client written from UTEST
63	Stop	Stop bit. This bit indicates the end the list of DCF Records. **NOTE:** The Erased state of flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list. 0 Not the end of the list. 1 End of the list.

The following structure must be present:

- The first record must be a start record.

- DCF records containing configuration data must immediately follow the start record with no blank records between—an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed.
- The end of the configuration records are indicated by the presence of a stop record.

Each record is 64 bits in length. Table 9-3 shows the DCF start record.

Warning

The start record must be placed at the beginning of the DCF area in UTEST flash memory to indicate to the device that the following data records must be processed.

Table 9-3. DCF start record

0x00 0:31	0x04 32:63
0x05AA55AF	0x00000000

During system boot, while the reset signal is asserted, the SSCM reads the device configuration records and writes the data to the appropriate internal module registers. Many operational aspects of the device are therefore already configured when the reset signal is released and normal device operation begins.

Some device configuration records are calculated during production testing and programmed into the TEST flash memory area. Other DCF records are developed by the user and programmed into the UTEST flash area.

The stop bit designates the last device configuration record in a set of records. It is not normally set in a valid UTEST DCF record programmed during production as this prevents additional UTEST records from being appended to the factory programmed UTEST DCF records. However, the flash memory location following the last UTEST DCF record programmed at the factory is an unprogrammed location whose contents is 0xFFFF_FFFF_FFFF_FFFF. The stop bit location in this unprogrammed flash memory location is therefore a logic 1, signifying that this is the last DCF record and not to be acted on.

The general format of the stop record is shown in Table 9-4. Only the stop bit needs to be 1 in order to form a stop record—all other bits are ignored. An unprogrammed location in UTEST flash is interpreted as a stop record.

Table 9-4. DCF stop record

0:31	32:62	63
Reserved	Reserved	1

If *n* data records are to be stored in UTEST, the data structure must be as shown in Table 9-5.

Table 9-5. Series of DCF records in UTEST flash memory

ADDR offset	DATA			
0x00	0x05AA55AF			
0x04	0x00000000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
…	…			
8n + 0x0	reserved			
8n + 0x4	Reserved			STOP=1
8(n+1) + 0x0	—			
8(n+1) + 0x4				

Warning

There must never be an unprogrammed record in the DCF data structure, as it is interpreted as a stop record and subsequent records are ignored. Records to be programmed in several sessions, appending new records to the end of the list each time, is shown in Figure 9-2

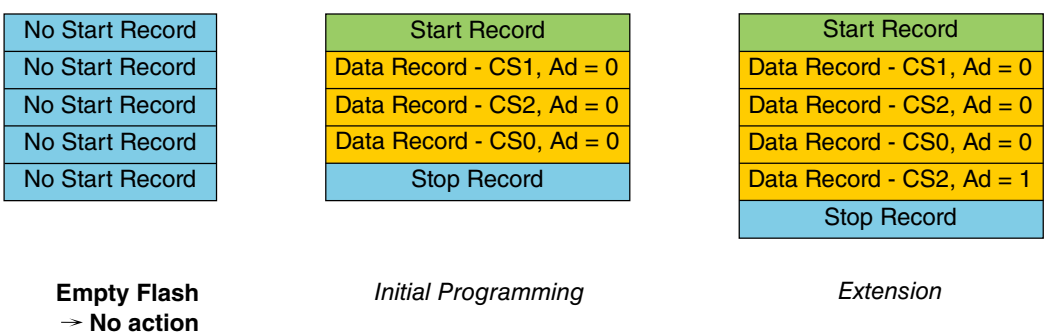


Figure 9-2. Appending DCF records

It is possible for more than one DCF Record to write to the same DCF client. The later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

9.3.1 UTEST DCF records

UTEST DCF records are located in the UTEST flash memory area (refer to UTEST flash memory map in the Memory map chapter). Some UTEST DCF records are programmed at the factory and the user may add USER DCF records from the first unprogrammed location following the factory programmed UTEST DCF records. An unprogrammed location in flash memory contains 0xFFFF_FFFF.

When programming UTEST DCF records, the records must start at the first address in the UTEST flash memory area and be continuous; that is, one UTEST DCF record must immediately follow the previous UTEST DCF record. An unprogrammed location in the UTEST flash memory area is interpreted by the SSCM as the end of the UTEST DCF records. When this happens, the SSCM passes control of the boot sequence to the Reset Generation Module.

9.4 DCF client table

A list of DCF clients is shown in DCF client list. Various attributes of the DCF clients are listed in these tables:

- DCF CS[14:0]

  The DCF Chip Select is a 15-bit field with each bit in the field acting as module-select signal for the internal modules. Only one bit in this field should be set in a DCF record.

- DCF address[16:2]

  The 15-bit ADDR field specifies an internal address for a DCF client. These addresses are only used by the SSCM when interpreting DCF records and writing data to specified modules.

- DCF client description

  The name of the DCF clients.

- Reset value of DCF client

  The reset value of DCF client before being written by the SSCM with the associated DCF record.

- IPS read

  This column specifies whether or not a DCF client can be read by user software after the release of reset.

- DCF client special strategy

  Registers are designated as DCF clients if they need to be written with a DCF record. Other features that must be designated for DCF clients include write strategies listed below. The following list defines the special strategies.

  - None

    No special DCF strategy is used.

  - Write Once

    A DCF client can be written only once. The DCF client ignores subsequent writes.

  - Triple Voted

    DCF clients have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. Triple voting allows for a 'bit-flip' error tolerance without changing the DCF client output data.

  - Write 0 only

    A bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a bit to logic 1 is ignored.

  - Write 1 only

    A bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a bit to logic 0 is ignored.

- DCF order dependency in flash memory

  This column details whether or not there are special conditions on the location of a particular DCF record in flash memory.

9.4.1 DCF client list

The DCF client list is provided in the following table:

Table 9-6. DCF client list

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
STCU - Support for 10 LBIST partitions; 78 Memories; 32-bit PRPG; 64-bit MISR						

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0000_0100	000000000000010	STCU_SKC	—	yes	None	First Record among STCU, Key1/Key2 dcf record sequence
000_0000_0000_0100	000000000000000	STCU_RUN		yes	None	None
000_0000_0000_0100	000000000000011	STCU_CFG		yes	None	None
000_0000_0000_0100	000000000000100	STCU_PLL_CFG	—	yes	None	None
000_0000_0000_0100	000000000000101	STCU_WDG	—	yes	None	None
000_0000_0000_0100	000000000000111	STCU_CRCE	—	yes	None	None
000_0000_0000_0100	000000000001010	STCU_ERR_FM	—	yes	None	None
000_0000_0000_0100	000000000001111	STCU_LBRMSW	—	yes	None	None
000_0000_0000_0100	000000000010000	STCU_LBUFM	—	yes	None	None
000_0000_0000_0100	000000000011101	STCU_MBUFML[31:0]	—	yes	None	None
000_0000_0000_0100	000000000011110	STCU_MBUFMM[31:0]	—	yes	None	None
000_0000_0000_0100	000000000011111	STCU_MBUFMH[31:0]	—	yes	None	None
000_0000_0000_0100	000000001000000	STCU_LB_CTRL_0	—	yes	None	None
000_0000_0000_0100	000000001000001	STCU_LB_PCS_0	—	yes	None	None
000_0000_0000_0100	000000001000010	STCU_LB_PRPGL_0	—	yes	None	None
000_0000_0000_0100	000000001000011	STCU_LB_PRPGH_0	—	yes	None	None
000_0000_0000_0100	000000001000100	STCU_LB_MISREL_0	—	yes	None	None
000_0000_0000_0100	000000001000101	STCU_LB_MISREH_0	—	yes	None	None
000_0000_0000_0100	000000001001000	STCU_LB_MISRELSW_0	—	yes	None	None
000_0000_0000_0100	000000001001001	STCU_LB_MISREHSW_0	—	yes	None	None
000_0000_0000_0100	000000001010000	STCU_LB_CTRL_1	—	yes	None	None
000_0000_0000_0100	000000001010001	STCU_LB_PCS_1	—	yes	None	None
000_0000_0000_0100	000000001010010	STCU_LB_PRPGL_1	—	yes	None	None
000_0000_0000_0100	000000001010011	STCU_LB_PRPGH_1	—	yes	None	None
000_0000_0000_0100	000000001010100	STCU_LB_MISREL_1	—	yes	None	None
000_0000_0000_0100	000000001010101	STCU_LB_MISREH_1	—	yes	None	None
000_0000_0000_0100	000000001011000	STCU_LB_MISRELSW_1	—	yes	None	None
000_0000_0000_0100	000000001011001	STCU_LB_MISREHSW_1	—	yes	None	None
000_0000_0000_0100	000000001100000	STCU_LB_CTRL_2	—	yes	None	None
000_0000_0000_0100	000000001100001	STCU_LB_PCS_2	—	yes	None	None
000_0000_0000_0100	000000001100010	STCU_LB_PRPGL_2	—	yes	None	None
000_0000_0000_0100	000000001100011	STCU_LB_PRPGH_2	—	yes	None	None
000_0000_0000_0100	000000001100100	STCU_LB_MISREL_2	—	yes	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0000_0100	000000001100101	STCU_LB_MISREH_2	—	yes	None	None
000_0000_0000_0100	000000001101000	STCU_LB_MISRELSW_2	—	yes	None	None
000_0000_0000_0100	000000001101001	STCU_LB_MISREHSW_2	—	yes	None	None
000_0000_0000_0100	000000001110000	STCU_LB_CTRL_3	—	yes	None	None
000_0000_0000_0100	000000001110001	STCU_LB_PCS_3	—	yes	None	None
000_0000_0000_0100	000000001110010	STCU_LB_PRPGL_3	—	yes	None	None
000_0000_0000_0100	000000001110011	STCU_LB_PRPGH_3	—	yes	None	None
000_0000_0000_0100	000000001110100	STCU_LB_MISREL_3	—	yes	None	None
000_0000_0000_0100	000000001110101	STCU_LB_MISREH_3	—	yes	None	None
000_0000_0000_0100	000000001111000	STCU_LB_MISRELSW_3	—	yes	None	None
000_0000_0000_0100	000000001111001	STCU_LB_MISREHSW_3	—	yes	None	None
000_0000_0000_0100	000000010000000	STCU_LB_CTRL_4	—	yes	None	None
000_0000_0000_0100	000000010000001	STCU_LB_PCS_4	—	yes	None	None
000_0000_0000_0100	000000010000010	STCU_LB_PRPGL_4	—	yes	None	None
000_0000_0000_0100	000000010000011	STCU_LB_PRPGH_4	—	yes	None	None
000_0000_0000_0100	000000010000100	STCU_LB_MISREL_4	—	yes	None	None
000_0000_0000_0100	000000010000101	STCU_LB_MISREH_4	—	yes	None	None
000_0000_0000_0100	000000010001000	STCU_LB_MISRELSW_4	—	yes	None	None
000_0000_0000_0100	000000010001001	STCU_LB_MISREHSW_4	—	yes	None	None
000_0000_0000_0100	000000010010000	STCU_LB_CTRL_5	—	yes	None	None
000_0000_0000_0100	000000010010001	STCU_LB_PCS_5	—	yes	None	None
000_0000_0000_0100	000000010010010	STCU_LB_PRPGL_5	—	yes	None	None
000_0000_0000_0100	000000010010011	STCU_LB_PRPGH_5	—	yes	None	None
000_0000_0000_0100	000000010010100	STCU_LB_MISREL_5	—	yes	None	None
000_0000_0000_0100	000000010010101	STCU_LB_MISREH_5	—	yes	None	None
000_0000_0000_0100	000000010011000	STCU_LB_MISRELSW_5	—	yes	None	None
000_0000_0000_0100	000000010011001	STCU_LB_MISREHSW_5	—	yes	None	None
000_0000_0000_0100	000000010100000	STCU_LB_CTRL_6	—	yes	None	None
000_0000_0000_0100	000000010100001	STCU_LB_PCS_6	—	yes	None	None
000_0000_0000_0100	000000010100010	STCU_LB_PRPGL_6	—	yes	None	None
000_0000_0000_0100	000000010100011	STCU_LB_PRPGH_6	—	yes	None	None
000_0000_0000_0100	000000010100100	STCU_LB_MISREL_6	—	yes	None	None
000_0000_0000_0100	000000010100101	STCU_LB_MISREH_6	—	yes	None	None
000_0000_0000_0100	000000010101000	STCU_LB_MISRELSW_6	—	yes	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0000_0100	000000010101001	STCU_LB_MISREHSW_6	—	yes	None	None
000_0000_0000_0100	000000010110000	STCU_LB_CTRL_7	—	yes	None	None
000_0000_0000_0100	000000010110001	STCU_LB_PCS_7	—	yes	None	None
000_0000_0000_0100	000000010110010	STCU_LB_PRPGL_7	—	yes	None	None
000_0000_0000_0100	000000010110011	STCU_LB_PRPGH_7	—	yes	None	None
000_0000_0000_0100	000000010110100	STCU_LB_MISREL_7	—	yes	None	None
000_0000_0000_0100	000000010110101	STCU_LB_MISREH_7	—	yes	None	None
000_0000_0000_0100	000000010111000	STCU_LB_MISRELSW_7	—	yes	None	None
000_0000_0000_0100	000000010111001	STCU_LB_MISREHSW_7	—	yes	None	None
000_0000_0000_0100	000000011000000	STCU_LB_CTRL_8	0x00000000	yes	None	None
000_0000_0000_0100	000000011000001	STCU_LB_PCS_8	—	yes	None	None
000_0000_0000_0100	000000011000010	STCU_LB_PRPGL_8	—	yes	None	None
000_0000_0000_0100	000000011000011	STCU_LB_PRPGH_8	—	yes	None	None
000_0000_0000_0100	000000011000100	STCU_LB_MISREL_8	—	yes	None	None
000_0000_0000_0100	000000011000101	STCU_LB_MISREH_8	—	yes	None	None
000_0000_0000_0100	000000011001000	STCU_LB_MISRELSW_8	—	yes	None	None
000_0000_0000_0100	000000011001001	STCU_LB_MISREHSW_8	—	yes	None	None
000_0000_0000_0100	000000011010000	STCU_LB_CTRL_9	—	yes	None	None
000_0000_0000_0100	000000011010001	STCU_LB_PCS_9	—	yes	None	None
000_0000_0000_0100	000000011010010	STCU_LB_PRPGL_9	—	yes	None	None
000_0000_0000_0100	000000011010011	STCU_LB_PRPGH_9	—	yes	None	None
000_0000_0000_0100	000000011010100	STCU_LB_MISREL_9	—	yes	None	None
000_0000_0000_0100	000000011010101	STCU_LB_MISREH_9	—	yes	None	None
000_0000_0000_0100	000000011011000	STCU_LB_MISRELSW_9	—	yes	None	None
000_0000_0000_0100	000000011011001	STCU_LB_MISREHSW_9	—	yes	None	None
000_0000_0000_0100	000000110000000	STCU_MB_CTRL_0	—	yes	None	None
000_0000_0000_0100	000000110000001	STCU_MB_CTRL_1	—	yes	None	None
000_0000_0000_0100	000000110000010	STCU_MB_CTRL_2	—	yes	None	None
000_0000_0000_0100	000000110000011	STCU_MB_CTRL_3	—	yes	None	None
000_0000_0000_0100	000000110000100	STCU_MB_CTRL_4	—	yes	None	None
000_0000_0000_0100	000000110000101	STCU_MB_CTRL_5	—	yes	None	None
000_0000_0000_0100	000000110000110	STCU_MB_CTRL_6	—	yes	None	None
000_0000_0000_0100	000000110000111	STCU_MB_CTRL_7	—	yes	None	None
000_0000_0000_0100	000000110001000	STCU_MB_CTRL_8	—	yes	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0000_0100	000000110001001	STCU_MB_CTRL_9	—	yes	None	None
000_0000_0000_0100	000000110001010	STCU_MB_CTRL_10	—	yes	None	None
000_0000_0000_0100	000000110001011	STCU_MB_CTRL_11	—	yes	None	None
000_0000_0000_0100	000000110001100	STCU_MB_CTRL_12	—	yes	None	None
000_0000_0000_0100	000000110001101	STCU_MB_CTRL_13	—	yes	None	None
000_0000_0000_0100	000000110001110	STCU_MB_CTRL_14	—	yes	None	None
000_0000_0000_0100	000000110001111	STCU_MB_CTRL_15	—	yes	None	None
000_0000_0000_0100	000000110010000	STCU_MB_CTRL_16	—	yes	None	None
000_0000_0000_0100	000000110010001	STCU_MB_CTRL_17	—	yes	None	None
000_0000_0000_0100	000000110010010	STCU_MB_CTRL_18	—	yes	None	None
000_0000_0000_0100	000000110010011	STCU_MB_CTRL_19	—	yes	None	None
000_0000_0000_0100	000000110010100	STCU_MB_CTRL_20	—	yes	None	None
000_0000_0000_0100	000000110010101	STCU_MB_CTRL_21	—	yes	None	None
000_0000_0000_0100	000000110010110	STCU_MB_CTRL_22	—	yes	None	None
000_0000_0000_0100	000000110010111	STCU_MB_CTRL_23	—	yes	None	None
000_0000_0000_0100	000000110011000	STCU_MB_CTRL_24	—	yes	None	None
000_0000_0000_0100	000000110011001	STCU_MB_CTRL_25	—	yes	None	None
000_0000_0000_0100	000000110011010	STCU_MB_CTRL_26	—	yes	None	None
000_0000_0000_0100	000000110011011	STCU_MB_CTRL_27	—	yes	None	None
000_0000_0000_0100	000000110011100	STCU_MB_CTRL_28	—	yes	None	None
000_0000_0000_0100	000000110011101	STCU_MB_CTRL_29	—	yes	None	None
000_0000_0000_0100	000000110011110	STCU_MB_CTRL_30	—	yes	None	None
000_0000_0000_0100	000000110011111	STCU_MB_CTRL_31	—	yes	None	None
000_0000_0000_0100	000000110100000	STCU_MB_CTRL_32	—	yes	None	None
000_0000_0000_0100	000000110100001	STCU_MB_CTRL_33	—	yes	None	None
000_0000_0000_0100	000000110100010	STCU_MB_CTRL_34	—	yes	None	None
000_0000_0000_0100	000000110100011	STCU_MB_CTRL_35	—	yes	None	None
000_0000_0000_0100	000000110100100	STCU_MB_CTRL_36	—	yes	None	None
000_0000_0000_0100	000000110100101	STCU_MB_CTRL_37	—	yes	None	None
000_0000_0000_0100	000000110100110	STCU_MB_CTRL_38	—	yes	None	None
000_0000_0000_0100	000000110100111	STCU_MB_CTRL_39	—	yes	None	None
000_0000_0000_0100	000000110101000	STCU_MB_CTRL_40	—	yes	None	None
000_0000_0000_0100	000000110101001	STCU_MB_CTRL_41	—	yes	None	None
000_0000_0000_0100	000000110101010	STCU_MB_CTRL_42	—	yes	None	None
000_0000_0000_0100	000000110101011	STCU_MB_CTRL_43	—	yes	None	None
000_0000_0000_0100	000000110101100	STCU_MB_CTRL_44	—	yes	None	None
000_0000_0000_0100	000000110101101	STCU_MB_CTRL_45	—	yes	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0000_0100	000000110101110	STCU_MB_CTRL_46	—	yes	None	None
000_0000_0000_0100	000000110101111	STCU_MB_CTRL_47	—	yes	None	None
000_0000_0000_0100	000000110110000	STCU_MB_CTRL_48	—	yes	None	None
000_0000_0000_0100	000000110110001	STCU_MB_CTRL_49	—	yes	None	None
000_0000_0000_0100	000000110110010	STCU_MB_CTRL_50	—	yes	None	None
000_0000_0000_0100	000000110110011	STCU_MB_CTRL_51	—	yes	None	None
000_0000_0000_0100	000000110110100	STCU_MB_CTRL_52	—	yes	None	None
000_0000_0000_0100	000000110110101	STCU_MB_CTRL_53	—	yes	None	None
000_0000_0000_0100	000000110110110	STCU_MB_CTRL_54	—	yes	None	None
000_0000_0000_0100	000000110110111	STCU_MB_CTRL_55	—	yes	None	None
000_0000_0000_0100	000000110111000	STCU_MB_CTRL_56	—	yes	None	None
000_0000_0000_0100	000000110111001	STCU_MB_CTRL_57	—	yes	None	None
000_0000_0000_0100	000000110111010	STCU_MB_CTRL_58	—	yes	None	None
000_0000_0000_0100	000000110111011	STCU_MB_CTRL_59	—	yes	None	None
000_0000_0000_0100	000000110111100	STCU_MB_CTRL_60	—	yes	None	None
000_0000_0000_0100	000000110111101	STCU_MB_CTRL_61	—	yes	None	None
000_0000_0000_0100	000000110111110	STCU_MB_CTRL_62	—	yes	None	None
000_0000_0000_0100	000000110111111	STCU_MB_CTRL_63	—	yes	None	None
000_0000_0000_0100	000000111000000	STCU_MB_CTRL_64	—	yes	None	None
000_0000_0000_0100	000000111000001	STCU_MB_CTRL_65	—	yes	None	None
000_0000_0000_0100	000000111000010	STCU_MB_CTRL_66	—	yes	None	None
000_0000_0000_0100	000000111000011	STCU_MB_CTRL_67	—	yes	None	None
000_0000_0000_0100	000000111000100	STCU_MB_CTRL_68	—	yes	None	None
000_0000_0000_0100	000000111000101	STCU_MB_CTRL_69	—	yes	None	None
000_0000_0000_0100	000000111000110	STCU_MB_CTRL_70	0x00000000	yes	None	None
000_0000_0000_0100	000000111000111	STCU_MB_CTRL_71	0x00000000	yes	None	None
000_0000_0000_0100	000000111001000	STCU_MB_CTRL_72	0x00000000	yes	None	None
000_0000_0000_0100	000000111001001	STCU_MB_CTRL_73	0x00000000	yes	None	None
000_0000_0000_0100	000000111001010	STCU_MB_CTRL_74	0x00000000	yes	None	None
000_0000_0000_0100	000000111001011	STCU_MB_CTRL_75	0x00000000	yes	None	None
000_0000_0000_0100	000000111001100	STCU_MB_CTRL_76	0x00000000	yes	None	None
000_0000_0000_0100	000000111001101	STCU_MB_CTRL_77	0x00000000	yes	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
PASS						
000_0000_0000_1000	000000000101000 - 000000000101011	Reserved				
000_0000_0000_1000	000000000101100	Censorship	0x0000	no	None	None
000_0000_0000_1000	000000000101101 – 000000000101111	Reserved				
000_0000_0000_1000	000000000110000	Production Disable	0x0000	no	Write Once	None
000_0000_0000_1000	000000000111000 – 000000000111111	Reserved				
000_0000_0000_1000	000000001000000	LOCK0_PG0	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000001	LOCK1_PG0	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000010	LOCK2_PG0	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000011	LOCK3_PG0	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000100	LOCK0_PG1	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000101	LOCK1_PG1	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000110	LOCK2_PG1	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001000111	LOCK3_PG1	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001000	LOCK0_PG2	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001001	LOCK1_PG2	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001010	LOCK2_PG2	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001011	LOCK3_PG2	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001100	LOCK0_PG3	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001101	LOCK1_PG3	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001110	LOCK2_PG3	0xFFFF_FFFF	yes	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0000_1000	000000001001111	LOCK3_PG3	0xFFFF_FFFF	yes	None	None
000_0000_0000_1000	000000001001100 – 000000011111111	Reserved				
Tamper Detect						
000_0000_0001_0000	000000000000000	Diary Base Address	0xFFFFFFFF	no	Write Once + Write '0' only	None
000_0000_0001_0000	000000000000001	Tamper Region Override	0x00	no	Write Once + Write '1' only	None
000_0000_0001_0000	000000000000010	Software Tamper Region Override Disable	0x00	no	Write Once + Write '1' only	None
000_0000_0001_0000	000000000000011 – 000000000000111	Reserved				
000_0000_0001_0000	000000000001000	OTP_EN0	0x0000_0000	no	Write '1' only	None
000_0000_0001_0000	000000000001001	OTP_EN1	0x0000_0000	no	Write '1' only	None
000_0000_0001_0000	000000000001010	OTP_EN2	0x0000_0000	no	Write '1' only	None
000_0000_0001_0000	000000000001011	OTP_EN3	0x0000_0000	no	Write '1' only	None
000_0000_0001_0000	000000000001100 – 000000000010011	Reserved				
000_0000_0001_0000	000_0000_0001_0100	TDR0_LOCK0	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_0101	TDR0_LOCK1	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_0110	TDR0_LOCK2	0x0000_0000	no	Write Once + Write '1' only	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0001_0000	000_0000_0001_0111	TDR0_LOCK3	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1000	TDR1_LOCK0	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1001	TDR1_LOCK1	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1010	TDR1_LOCK2	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1011	TDR1_LOCK3	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1100	TDR2_LOCK0	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1101	TDR2_LOCK1	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1110	TDR2_LOCK2	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0001_1111	TDR2_LOCK3	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0000	TDR3_LOCK0	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0001	TDR3_LOCK1	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0010	TDR3_LOCK2	0x0000_0000	no	Write Once + Write '1' only	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_0001_0000	000_0000_0010_0011	TDR3_LOCK3	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0100	TDR4_LOCK0	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0101	TDR4_LOCK1	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0110	TDR4_LOCK2	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_0111	TDR4_LOCK3	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_1000	TDR5_LOCK0	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_1001	TDR5_LOCK1	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_1010	TDR5_LOCK2	0x0000_0000	no	Write Once + Write '1' only	None
000_0000_0001_0000	000_0000_0010_1011	TDR5_LOCK3	0x0000_0000	no	Write Once + Write '1' only	None
MISCELLANEOUS						
000_0000_1000_0000	000000000000000	UTEST Miscellaneous(See Miscellaneous DCF registers x for bit position details.)	0x00048941	no	Triple Voted	None
000_0000_1000_0000	000000000000101 – 000000000000011	Reserved				
000_0000_1000_0000	00000000000100	IVPR0	0x5000_0000	no	None	None

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
000_0000_1000_0000	00000000000101	IVPR1	0x5100_0000	no	None	None
000_0000_1000_0000	00000000000110	IVPR2	0x5200_0000	no	None	None
000_0000_1000_0000	00000000000111	JTAG Pin Config	0x0000_0001	no	None	None
000_0000_1000_0000	000000000001000 – 000000000011110	Reserved				
000_0000_1000_0000	000000000011111	PMC_REE_BUS	0x0000_0000	no	Triple Voted	None
000_0000_1000_0000	000000000100000-000000000110101	Reserved				
000_0000_1000_0000	000000000110110	PMC VREG1P2_ENB	0x0000_0001	no	Triple Voted	None
000_0000_1000_0000	000000000110111-000000011111111	Reserved				
BAF 'Soft' Clients						
100_0000_0000_0000	0x0000	Security Watchdog Control Register	NA	no	Write Once	None
100_0000_0000_0000	0x0001	Security Watchdog Timeout	NA	no	Write Once	None
100_0000_0000_0000	0x0002	Security Watchdog Service Address	NA	no	Write Once	None
100_0000_0000_0000	0x0003	Security Watchdog CPU select	NA	no	Write Once	None
100_0000_0000_0000	0x0004 – 0x01FF	Reserved	NA			
100_0000_0000_0000	0x0200	Address	NA	no	None	Must be written first as a pair with a data register
100_0000_0000_0000	0x0201	32-bit Data	NA	no	None	Must be written second as a pair with the Address register
100_0000_0000_0000	0x0202	16-bit Data	NA	no	None	Must be written second as a pair with the Address register

Table continues on the next page...

Table 9-6. DCF client list (continued)

DCF CS[14:0][1]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	IPS read	DCF client special strategy	DCF order dependency in flash memory
100_0000_0000_0000	0x0203	8-bit Data	NA	no	None	Must be written second as a pair with the Address register
100_0000_0000_0000	0x0204 – 0x020F	Reserved	NA			
100_0000_0000_0000	0x0210	Callback address	NA	no	None	None
100_0000_0000_0000	0x0211 – 0x7FFF	Reserved	NA			

1. DCF programming should start from address 0x00400208.

9.4.2 Miscellaneous DCF registers

The DCF registers are as follows:

Table 9-7. UTEST Miscellaneous DCF Client

Offset: 000000000000000									Access: DCF Client							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													XOSC_IBIAS_SEL		
W	Reserved															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	OPT_RGM_ESR0_HW	XOSC_ ALC_DIS	XOSC_ LF_EN	Reserved	Reserved	OSC_ EN_40MHZ	Reserved	LOCKSTEP_ EN	XOSC_ EN	XOSC_ EXT_ CLOAD	XOSC_LOAD_CAP_SEL[4:0]					ESR0 GATE
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Table 9-8. UTEST Miscellaneous DCF Client descriptions

Field	Description
0 - 12 Reserved	Reserved.
13 - 15	XOSC_IBIAS_SEL

Table continues on the next page...

Table 9-8. UTEST Miscellaneous DCF Client descriptions (continued)

Field	Description
XOSC_IBIAS_SEL	These 3 bits control the XOSC bias current. 1x is the default current consumption in the external oscillator (XOSC): 000 0 (no current) 001 0.25x 010 0.5x 011 Reserved 100 1x 101 1.25x 110 1.5x 111 1.75x (maximum current)
16 OPT_RGM_ESR0_HW	OPT_RGM_ESR0_HW. 0 Allows MC_RGM to continue to assert ESR0 until RGM_EROEC[EROEC] bit has been cleared by software. 1 Does not allow MC_RGM to continue to assert ESR0 until RGM_EROEC[EROEC] bit has been cleared by software.
17 XOSC_ALC_DIS	XOSC_ALC_DIS 0 Enable Automatic Level Controller 1 Disable Automatic Level Controller
18 XOSC_LF_EN	XOSC_LF_EN **NOTE:** It is possible to use a low-frequency oscillator (less than 16 MHz) to configure properly the CMU_PLL instance by modifying CSR[RCDIV]. To configure properly the field RCDIV, see CLKMN0_RMT supervisor. 0 Oscillator is not used for low frequency (~8 MHz) 1 Oscillator is used for low frequency (~8 MHz)
19	Reserved
20	Reserved
21 OSC_EN_40MHZ	Select external 40MHz oscillator 0 20 MHz 1 40 MHz XOSC
22	Reserved
23 LOCKSTEP_EN	LOCKSTEP_EN — Enable Safety Core to be in lockstep with Main Core 0 Lockstep disabled 1 Lockstep enabled
24 XOSC_EN	XOSC_EN — Enable XOSC Determines value of XOSCON bit in MC_ME_RESET_MC and MC_ME_DRUN_MC registers. 0 XOSCON bit cleared. XOSC disabled. 1 XOSCON bit set. XOSC enabled.
25 XOSC_EXT_CLOAD	XOSC_EXT_CLOAD 0 Selects XOSC internal cap (ok for 20–40 MHz operation) 1 External (required for 4–16 MHz operation)
26 - 30 XOSC_LOAD_CAP_ SEL[0:4]	XOSC_LOAD_CAP_SEL [0:4] — Five trim bits to program the load capacitance. Load capacitor values are determined by the crystal manufacturer data sheet. See XOSC Start-up for more information.

Table continues on the next page...

Table 9-8. UTEST Miscellaneous DCF Client descriptions (continued)

Field	Description
31 ESR0_GATE	ESR0_Gate — Gate ESR0 Input Pin 0 ESR0 input disabled (gated). ESR0 only asserted as an output from the device. 1 ESR0 input enabled (not gated). Assertion of the $\overline{\text{ESR0}}$ pin causes a hard system reset.

Table 9-9. PMC_REE bits DCF Client

Offset: 000000000011111												Access: DCF Client				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	Reserved															
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	HVD8_F	LVD10_A	HVD8_C	HVD7_F	HVD15_A	LVD3_P	HVD15_C	LVD14_A	LVD14_IM	Reserved	LVD13_IM	LVD10_F	HVD7_C	LVD4_C	LVD3_F	LVD3_C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-10. PMC_REE bits DCF client descriptions

Field	Description
0 – 15	Reserved
16	Reserved
17 LVD10_A	LVD10_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
18 HVD8_C	HVD8_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD8[HVD8_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.
19 HVD7_F	HVD7_F reset enable. This bit defines whether an HVD assertion on the supply of the flash generates a system reset. The RES_VD7[HVD7_F] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the flash does not cause system reset. 1 Enabled. HVD assertion on the supply of the flash causes system reset.

Table continues on the next page...

Table 9-10. PMC_REE bits DCF client descriptions (continued)

Field	Description
20 HVD15_A	HVD15_A reset enable. This bit defines whether an HVD assertion on the first supply of the ADC generates a system reset. The RES_VD15[HVD15_A] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the first supply of the ADC does not cause system reset. 1 Enabled. HVD assertion on the first supply of the ADC causes system reset.
21 LVD3_P	LVD3_P reset enable. This bit defines whether an LVD assertion on the supply of the PLL generates a system reset. The RES_VD3[LVD3_P] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the PLL does not cause system reset. 1 Enabled. LVD assertion on the supply of the PLL causes system reset.
22 HVD15_C	HVD15_C reset enable. An HVD assertion on the core high-voltage supply generates a system reset. RES_VD15[HVD15_C] controls whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the core high voltage does not cause system reset. 1 Enabled. HVD assertion on the supply of the core high voltage causes system reset.
23 LVD14_A	LVD14_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. The RES_VD14[LVD14_A] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
24 LVD14_IM	LVD14_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O Main generates a system reset. The RES_VD14[LVD14_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O Main does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O Main causes system reset.
25	Reserved
26 LVD13_IM	LVD13_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a system reset. The RES_VD13[LVD13_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes system reset.
27 LVD10_F	LVD10_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
28 HVD7_C	HVD7_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD7[HVD7_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

Table continues on the next page...

Table 9-10. PMC_REE bits DCF client descriptions (continued)

Field	Description
29 LVD4_C	LVD4_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD4[LVD4_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
30 LVD3_F	LVD3_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. The RES_VD3[LVD3_F] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
31 LVD3_C	LVD3_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD3[LVD3_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

Chapter 10
Power management

10.1 Overview

MPC5777M microcontrollers include a robust power management infrastructure that enables applications to select among various operational and low-power modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave reset. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Clock Generation Module (MC_CGM)
- Mode Entry Module (MC_ME)

This chapter describes these modules in brief. For complete details, see the respective module chapters.

The MC_CGM generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement low power modes.

The MC_PCU provides a status register and a submodule, the PMC_dig (Power Management Control Digital Interface), provides an interface to configure and control internal voltages.

The MC_ME module controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.

The MPC5777M has two types of modes, SYSTEM modes and USER modes.

SYSTEM modes:

- SYSTEM modes are entered automatically on system events or on application request.
- The SYSTEM modes are: RESET, SAFE, TEST and DRUN.

Table 10-1. SYSTEM modes

Mode	Description	Entry	Exit
RESET	System remains in this mode until all resources are available for S/W to take device control. RESET manages H/W initialization of chip configuration, voltage regulators, oscillators, PLLs, and flash.	System reset assertion from RGM	System reset deassertion from RGM
SAFE	May be entered on detection of a recoverable error. Forces system into predefined safe configuration from which the system may try to recover.	H/W fail, S/W request from DRUN, TEST and RUNx	System reset assertion, DRUN through S/W
TEST	Control environment for device self-test. Allows application to run its own self-test like flash checksum and RAM BIST.	Software request from DRUN	System reset asserted, DRUN through S/W
DRUN	Entry mode for the embedded software. Provides full system accessibility and enables system configuration. Provides gate to enter USER modes. BAF, when present, is executed in DRUN mode.	Reset removed, from SAFE, TEST and RUNx through S/W, STANDBY wakeup	Reset applied, RUNx, TEST through S/W, SAFE through S/W or H/W fail

USER modes:

- USER modes control performance and consumption during normal application operation. The modes are organized according to the performance provided.
- The USER modes are RUNx, HALT, STANDBY, and STOP.
- Each mode is entered by writing a dedicated register protected by a key, to avoid unwanted transition.
- Figure 10-1 shows the mode relationships.

Table 10-2. USER modes

Mode	Description	Entry	Exit
RUNx	These are software running modes where most of the processing activity is done. These run modes allow the user to enable different clock and power configurations of the system with respect to each other.	Software request from DRUN, interrupt event from HALT, wakeup request from STOP	Reset asserted, SAFE through S/W or H/W fail, RUNx, HALT, STOP, STANDBY through S/W
HALT	Reduced-activity low-power mode during which the core clock is disabled. It can be configured to switch off analog peripherals like PLL, flash, main voltage regulator, etc., for efficient power management at the cost of higher wakeup latency.	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on interrupt event

Table continues on the next page...

Table 10-2. USER modes (continued)

Mode	Description	Entry	Exit
STOP	Advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including the oscillator for efficient power management at the cost of higher wakeup latency.	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on wakeup event
STANDBY	Reduced-leakage low-power mode during which power is cut off from most of device. Wakeup takes a relatively long time, and content is lost or must be restored from backup	Software request from RUNx modes	Reset asserted, INIT on wakeup event

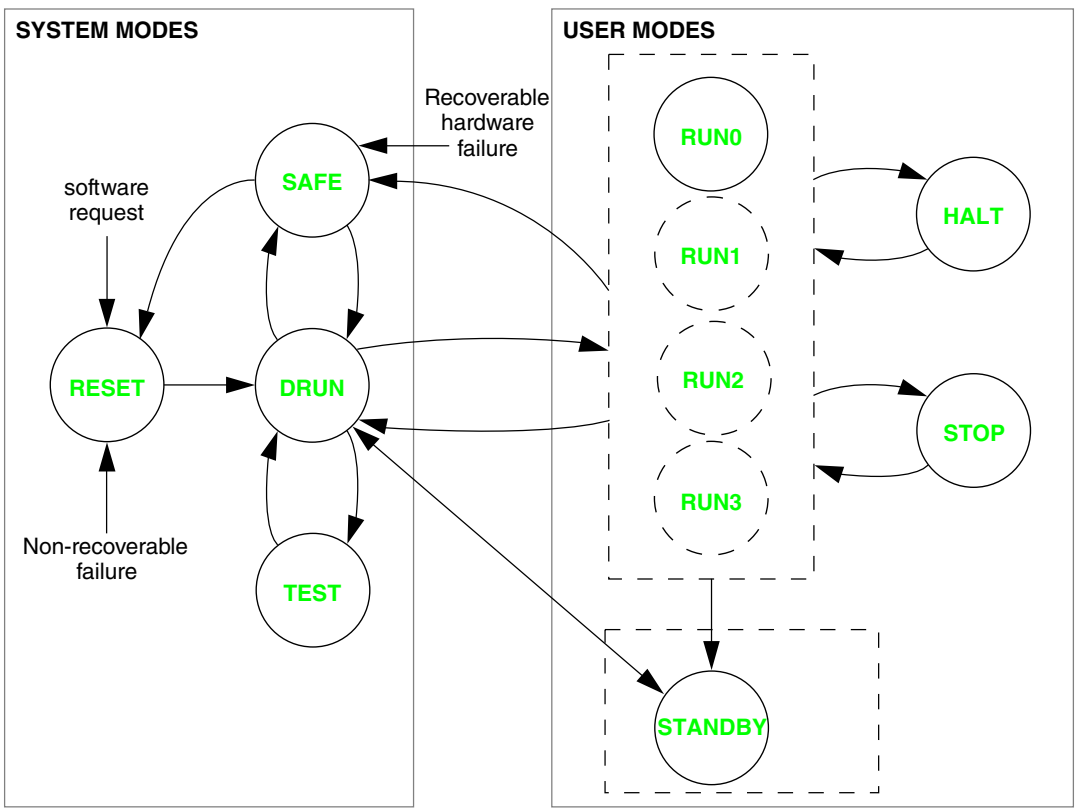


Figure 10-1. Mode relationships

10.1.1 Power management framework

The power management framework supports a variety of configuration options. See the following figure for more details. MPC5777M implements a unique core voltage power domain. There are no separate functional power domains.

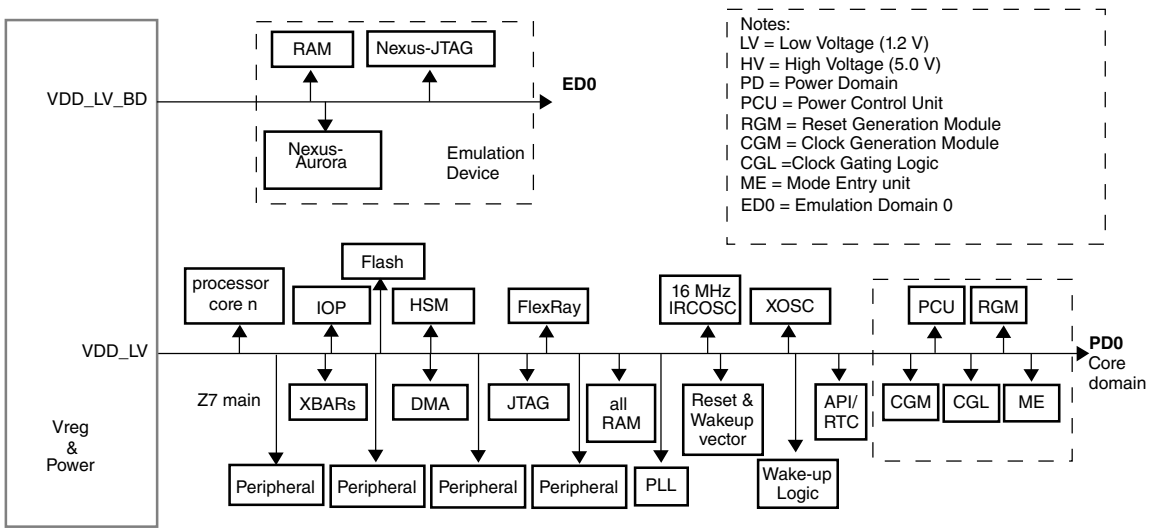


Figure 10-2. Device power management framework

10.1.2 Power management supply description

The following table provides an overview of the PMC supply signals.

Table 10-3. Device power management controller external signals

Name	Type	Description
VDD_HV_ADR_D	Reference	Voltage reference of ADC Σ/δ module
VDD_HV_ADR_S	Reference	Voltage reference of ADC SAR module
VDD_HV_ADV	Supply	High voltage supply for the ADC modules
VDD_HV_FLA	Decoupling	Decoupling supply pin for Flash
VDD_HV_IO	Supply	High voltage power supply for the I/Os
VDD_HV_IO_FLEX	Supply	FlexRay/Ethernet 3.3 V I/O supply
VDD_HV_IO_JTAG	Supply	Oscillator and JTAG pin supply
VDD_HV_PMC	Supply	High voltage power supply for internal power management unit
VDD_LV	Supply	Low voltage power supply for the core area
VDD_LV_BD	Supply	Low voltage power supply for the buddy device
VSS	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV in case of exposed pad device.
VSS_HV_ADR_D	Reference	Ground reference of ADC Σ/δ module
VSS_HV_ADR_S	Reference	Ground reference of ADC SAR module
VSS_HV_ADV	Supply	Ground supply for the ADC modules

Warning

There are constraints on some input voltages relative to other input voltages. Please refer to the device data sheet for detailed information.

10.1.3 MPC5777M power management controller overview

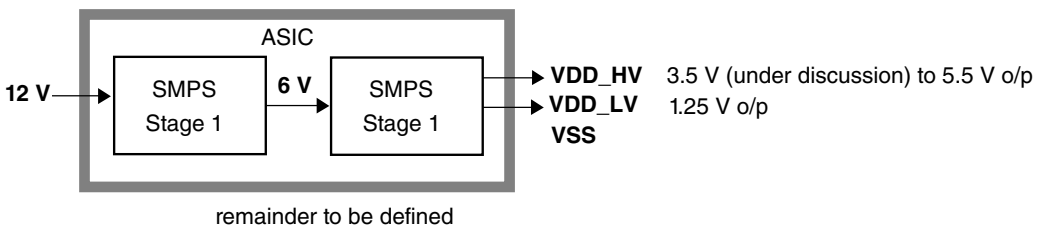


Figure 10-3. PMU MPC5777M (simplified)

VDD_HV_IO is redundant, ensuring that single pin failure does not prevent device from functioning correctly. Full specification may not be guaranteed.

VDD_LV is redundant, ensuring that single pin failure does not prevent the device from functioning correctly.

10.1.3.1 1.2 V safety/spike regulator specification

MPC5777M implements a 1.2 V safety/spike regulator. This regulator can be used to handle fast application current variation that cannot be fully handled by external regulator.

10.2 Low power mode support

The MPC5777M microcontroller supports low power modes.

Dedicated logic prevents uncontrolled activation of low-power mode. This is implemented using protected sequence from ME.

Any register leading to low power entry is triple-voted.

10.2.1 Low power support and STOP mode implementation

Low power mode may be used to control the engine during the power-up phase, as in a Start-Stop system or when the key has been turned off.

The device implements the equivalent of an idle or power-down mode through STOP and HALT mode configuration.

The microcontroller application code can be used to put the device or modules into STOP mode.

A combination of mode control and clock control allows the user to configure the STOP mode.

In Afterrun mode, peripheral modules can be enabled/disabled in STOP using the control described in the ME section. For alternate configurations, other modules can be enabled/disabled in STOP using the control described in the ME section. External interrupt request channels and external wake-up channels can be made active to wake-up the device in alternative to peripherals interrupts.

All pins associated to the active modules are functional.

Control of the pins is defined by user during run time. A wake-up line is associated to each selectable module.

One of the available modules can be configured to perform the wake-up task. For example, with five CAN modules on the chip, four modules are set completely to power down mode and only one is configured for wake-up message receive (any of the five modules can be configured to perform this task).

All CAN modules can be independently configured to receive messages and respond to wake-up messages. Control of the pins is defined by user during run time.

In idle mode these modules can also be idle, but able to wake-up as soon as activity is detected and wake the core.

All these modules can be configured to a low power state. Wake-up recovery is programmable, for instance depending on state or activity of associated pin.

All parts of the device can be disabled. Following modules are configurable:

- There is one CAN module, selectable from any of the CAN modules on the device. The others can be disabled.
    - The maximum CAN baud rate is 1 MBaud.
    - One CAN module is able to receive messages from the CAN bus and wake-up the controller if the received message contains the correct message identifier.

- One LIN/UART module is able to recognize special wake-up messages on the receive line and then wake up the controller.
- Some "request" pins are able to initiate a wake-up. Refer to the pinout chapter for more information.
- The system timer is able to initiate a wake-up.

Configurable modules (ASC, three CANs, timer) can all be off in STOP mode.

The CAN module recovers from wake-up instantaneously but its ability to capture the message depends on settings of certain peripherals such as the CAN protocol clock. Depending on recovery time this impacts whether the first message can be captured, hence software must be used as a workaround.

Device implements all three mechanisms to exit from stop mode: pin wakeup, CAN wake-up and timed wake-up.

10.2.1.1 After-run implementation

After-run mode is implemented mainly making use of clock gating, module disabling, reduced frequency, mode switching.

Assuming that the selected CAN is kept enabled (clocked and active) then it can maintain its synchronization with the CAN network. Assuming also that the supplied CAN module can buffer the received (first) message, the CAN is then able to interrupt the IOP. The IOP would execute from TCM (i.e., Flash is disabled) and use the internal RC oscillator (IRCOSC) or the external oscillator (XOSC). To support this would keep the PMC, LVD, temperature sensor, internal FIRC, XOSC and pads active. The PLL, Flash, and ADC would be disabled, with rest of part static and clock gated off.

10.2.1.2 Checker core disabling

The checker core is always in lock step with the main core — hence if the main core is clocked, the checker core is clocked (unless user intentionally disables checker clock, e.g. afterrun mode). Also if the main core is gated off, then the checker core is also gated off.

It is possible to disable the safety checker core to reduce current consumption.

The checker core can be placed in idle mode throughout the afterrun mode. By default the checker core mimics the main core, but it can be clock gated off during afterrun mode whilst continuing to clock the main core.

The checker core is implemented in such a way that its entire clock tree can be gated off while the core logic remains supplied.

The checker core (part of the Checker Logic) and the corresponding 'checked main' core share the same cache SRAM arrays and local SRAM arrays. Once the checker logic is disabled (i.e., its clock is gated) the SRAM array of the cache and the local memory of the main core are still usable by the main core.

10.3 Flash power requirements

During start-up operation, the flash module operates from an untrimmed regulator, which has a slightly larger variation than the ideal case. During this start-up phase, the flash can still be read, but it must be read at a slower rate. For MPC5777M, read is performed differentially, i.e., the data is duplicated / inverted across two locations. While operating in this slower read mode, the flash can operate at a much reduced voltage, and hence the untrimmed regulator should default to a < 3.3V average. This will help guarantee that the maximum value is not exceeded.

Following completion of the start-up phase, the regulator is in its trimmed condition, and subsequently the flash module reference current has also been trimmed. Therefore, when entering normal operation, the flash can be read in the normal access manner and support the full flash specification.

For MPC5777M, the flash is using the externally supplied 1.25 V for the low voltage supply, while high voltage (3.3 V) is generated by a dedicated internal regulator whose supply is VDD_HV_PMC. When the VDD_HV_PMC is below 3.5V, this regulator is in its bypass mode and can only support flash read.

10.4 Device trimming

During the initialization phase the device defaults to a pre-determined state for each of the LVDs, HVDs and the internal regulators. As the flash becomes available, the differential read process allows the trimmed data to be available for trimming the internal LVDs, HVDs and regulators.

Please refer to Power sequence for further details.

10.5 Supply monitoring (POR and LVDs)

The function of the POR and LVD circuits is to hold the device in reset regardless of how slow the supply voltage rise is, until the point at which the POR and LVDs are released.

The POR and LVD circuits function correctly even if the input voltage is non-monotonic.

10.5.1 Power-on reset (POR)

The PMC implements two internal power-on reset circuits:

- VPORUPLV monitors the voltage on the 1.2 V input supply. It is monitoring the VDD_LV_CORE pin. The VPORUPLV asserts a reset when the input supply is below defined values.
- VPORUPHV monitors the voltage on the 5.0 V input supply. It is monitoring the VDD_HV_PMC pin. The POR trip point is high enough to make sure all the LVD circuits are functional.

See Reset and Boot chapter for PORST/ESR0 pin functionality.

10.5.2 Behavior of device LVD / HVD

Although there is an option to disable the LVD and HVD following reset (see Low Voltage Detection (LVD) and High Voltage Detection (HVD)), they are capable of being used in a 'monitor' only mode and also capable of generating a safe / interrupt event. The diagram in the following figure illustrates the behavior of these internal LVD / HVD circuits during the power on phase, through into device initialization and subsequently in 'normal' run / operating mode.

The LVDs/HVDs can also be configured after device initialization preventing reset to happen when supply crosses the LVD threshold, effectively providing a higher voltage operating range. It is the responsibility of the application to ensure that the device remains in the functional range.

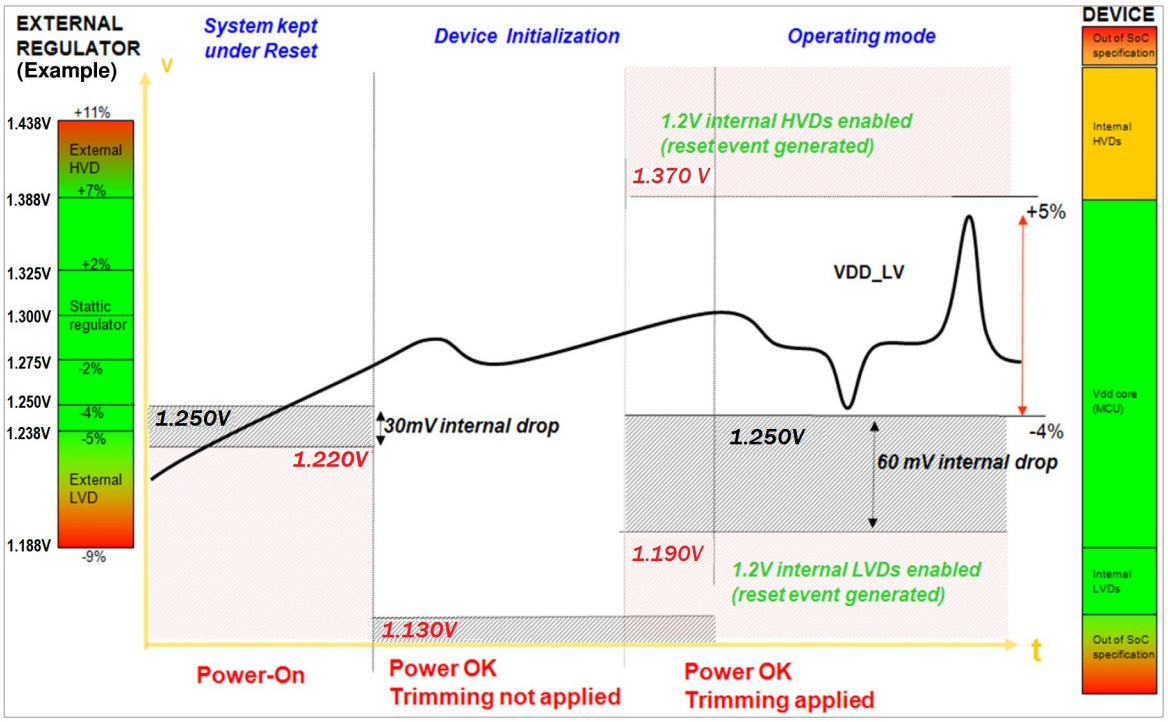


Figure 10-4. Internal LVD and HVD in configuration example during power-up

10.5.3 Low Voltage Detection (LVD) and High Voltage Detection (HVD)

The internal LVD circuits monitor when the voltage on the corresponding supply is below defined values and either assert a reset, an interrupt, or set a flag in a register. The LVDs also support hysteresis in the falling and rising trip points.

10.5.3.1 LVD and HVD implementation

- All LVDs and HVDs are capable of generating either an RGM and/or FCCU event, with the following exceptions:
  - The LVD270 (that is, the LVD monitoring the high voltage supply input) should be disabled only for test purpose and should otherwise always generate a reset when triggered.

WARNING

In case LVD270 are disabled, the device may not detect through internal monitor that VDD_HV voltage is outside functional range. If this event occurs, reset state must be ensured by assertion of external PORST pin from the user.

  - It is not possible to disable the LVD096 (that is, the LVD monitoring the internal core voltage) and they always generate a reset when triggered. This LVD096 is integral to the POR management.

- All LVDs and HVD configured for reset generation can cause functional or destructive resets. MC_RGM PHASE0 is not exited until all destructive reset conditions are cleared.
- The appropriate flag bits in the PMC registers are set by LVD and HVD events.
- The LVDs and HVDs control is protected by the SoC wide register protection scheme. Therefore it is configurable as long as the scheme is followed. Please refer to Chapter 81, Register Protection (REG_PROT).
- MPC5777M implements REE DCF records to allow to associated "configurable" LVDs/HVDs to a RESET event. When RESET event is enabled through REE DCF record, it cannot be changed until next RESET event. When RESET event is not enabled through REE DCF record, LVDs/HVDs trigger event default to FCCU event.

Note

The list of the "configurable" LVDs/HVDs is provided in Table 10-4 , in which the "User configurability" columns provide the possible user configuration.

- When the LVD or the HVD is enabled for destructive reset generation and a subsequent trigger event is detected, the external PORST pin is driven low.

Note

LVD096 and LVD270 (low range LVDs) cannot be disabled by DCF bits written by the user during boot sequence. These modules are used during power-up phase and must ensure that an absolute lowest threshold of operation is never crossed. This

is not a guarantee that the device will function down to this level. It is rather a guarantee that the device will recover if this level is crossed.

WARNING

The LVD270 can be disabled for test purpose through software accessing PMC_REE register. In case LVD270 are disabled, the device may not detect through internal monitor that VDD_HV voltage is outside functional range (below LVD270 threshold). If this event occurs, reset state must be ensured by assertion of external PORST pin from the user. When supply goes below the threshold of the POR240 (internal power-on state detector), device will be forced into reset and LVD270 is automatically re-enabled.

Table 10-4. POR and voltage monitors description

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP-PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/FEE/ IE[1](RUN)
POWER MANAGEMENT UNIT VOLTAGE MONITORS – LOW VOLTAGE SUPPLY							
LVD096_C	VD2	Low voltage supply core low range low voltage detector	VDD_LV_CORE (hot-point)	ENABLE	ENABLE	ENABLE	ENABLE
LVD096_F	VD2	Low voltage supply flash low range low voltage detector	VDD_LV_FLA (hot-point)	ENABLE	ENABLE	ENABLE	ENABLE
LVD108_C	VD3	Low voltage supply core medium range low voltage detector	VDD_LV_CORE (hot-point)	DISABLE	DISABLE	LVDM[0]	VD3_C
LVD108_F	VD3	Low voltage supply flash medium range low voltage detector	VDD_LV_FLA (hot-point)	DISABLE	DISABLE	LVDM[1]	VD3_F
LVD108_P	VD3	Low voltage supply PLL medium range low voltage detector	VDD_LV_PLL (hot-point)	DISABLE	DISABLE	LVDM[10]	VD3_P
LVD112_C	VD4	Low voltage supply core medium range low voltage detector	VDD_LV_CORE (cold-point)	ENABLE	DISABLE	LVDM[2]	VD4_C
HVD140_C	VD7	Low voltage supply core high voltage detector	VDD_LV_CORE (cold-point)	DISABLE	DISABLE	LVDM[3]	VD7_C
HVD145_C	VD8	Low voltage supply core high voltage detector	VDD_LV_CORE (cold-point)	ENABLE	ENABLE	LVDM[13]	VD8_C

Table continues on the next page...

Table 10-4. POR and voltage monitors description (continued)

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP-PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/FEE/ IE[1](RUN)
HVD145_F	VD8	Low voltage supply flash high voltage detector	VDD_LV_FLA (hot-point)	DISABLE	DISABLE	LVDM[15]	VD8_F
POWER MANAGEMENT UNIT VOLTAGE MONITORS – HIGH VOLTAGE SUPPLY							
LVD270_C	VD9	High voltage supply core low range low voltage detector	VDD_HV_PMC (hot-point)	ENABLE	ENABLE	ENABLE	VD9_C
LVD270_F	VD9	High voltage supply flash low range low voltage detector	VDD_HV_FLA (hot-point)	ENABLE	ENABLE	ENABLE	VD9_F
LVD270_IM	VD9	High voltage supply I/O main low range low voltage detector	VDD_HV_IO_MAIN (hot-point)	ENABLE	ENABLE	ENABLE	VD9_IM
LVD270_IF	VD9	High voltage supply I/O FlexRay low range low voltage detector	VDD_HV_IO_FLEX (hot-point)	DISABLE	DISABLE	ENABLE	VD9_IF
LVD270_IF2	VD9	High voltage supply I/O FlexRay2 low range low voltage detector	VDD_HV_IO_FLEX 2 (hot-point)	DISABLE	DISABLE	ENABLE	VD9_IF2
LVD270_IJ	VD9	High voltage supply I/O JTAG low range low voltage detector	VDD_HV_IO_JTAG (hot-point)	ENABLE	ENABLE	ENABLE	VD9_IJ
LVD270_EBI	VD9	High voltage supply I/O oscillator low range low voltage detector	VDD_HV_IO_EBI (hot-point)	ENABLE	ENABLE	ENABLE	VD9_IE
LVD270_O	VD9	High voltage supply I/O oscillator low range low voltage detector	VDD_HV_IO_JTAG (hot-point)	ENABLE	ENABLE	ENABLE	VD9_O
LVD295_F	VD10	High voltage supply flash medium range low voltage detector	VDD_HV_FLA (hot-point)	ENABLE	ENABLE	LVDM[4]	VD10_F
LVD295_A	VD10	High voltage supply flash medium range low voltage detector	VDD_HV_ADV (hot-point)	ENABLE	ENABLE	LVDM[14]	VD10_A
HVD360_F	VD12	High voltage supply flash low range low voltage detector	VDD_HV_FLA (hot-point)	ENABLE (security)	ENABLE (security)	ENABLE	VD12_F
LVD360_IM	VD13	High voltage supply I/O main medium range low voltage detector	VDD_HV_IO_MAIN (hot-point)	ENABLE	DISABLE	LVDM[5]	VD13_IM
LVD400_IM	VD14	High voltage supply I/O main high range low voltage detector	VDD_HV_IO_MAIN (hot-point)	ENABLE	DISABLE	LVDM[7]	VD14_IM

Table continues on the next page...

Table 10-4. POR and voltage monitors description (continued)

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP-PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/FEE/ IE[1](RUN)
LVD400_A	VD14	High voltage supply ADC high range low voltage detector	VDD_HV_ADV (hot-point)	DISABLE	DISABLE	LVDM[8]	VD14_A
HVD600_C	VD15	High voltage supply PMC main high voltage detector	VDD_HV_PMC (cold-point)	ENABLE[2]	ENABLE [2]	LVDM[9]	VD15_C
HVD600_A	VD15	High voltage supply ADC main high voltage detector	VDD_HV_ADV (cold-point)	DISABLE	DISABLE	LVDM[11]	VD15_A

1. REE: Reset event enable / FEE: FCCU event enable / IE: Interrupt enable — See the PMC_dig chapter.
2. If this Voltage Detect is asserted, it will prevent exit from the current reset phase.

Table 10-5 provides the monitor status depending on configuration.

Table 10-5. Voltage monitors configurability

POWERUP-PHASE0	PHASE1-2	DCF	REE[1]	RES[2]	FEE[3]	IE[4]	Monitor status
ENABLE	ENABLE	—	—	—	—	—	**Default configuration (boot threshold monitor)** Monitor enabled during full run from power-up to power-down
ENABLE	DISABLE	ENABLE	ENABLE	DEST	—	—	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor re-enabled during phase 3 on DCF record reading. Trigger destructive reset event during reset flow. Configuration is locked to enable during run time.
ENABLE	DISABLE	ENABLE	ENABLE	FUNC	ENABLE	—	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor re-enabled during phase 3 on DCF record reading. Trigger destructive reset event during reset flow. Configuration is locked to enable during run time. **After reset completion:**

Table continues on the next page...

Table 10-5. Voltage monitors configurability (continued)

POWERUP-PHASE0	PHASE1-2	DCF	REE[1]	RES[2]	FEE[3]	IE[4]	Monitor status
							LVD event demoted from destructive reset to functional reset by software after completion of reset sequence. FCCU event are generated on functional reset.
ENABLE	DISABLE	ENABLE	ENABLE	FUNC	DISABLE	—	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor re-enabled during phase3 on DCF record reading. Trigger destructive reset event. Configuration is locked to enabled during run time. **After reset completion:** LVD event demoted from destructive reset to functional reset by software after completion of reset sequence. FCCU events are not generated.
ENABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	ENABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor disabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] The LVD event is also associated to an interrupt event.
ENABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	DISABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor disabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] Configuration can be changed by software to any of following events: functional reset, destructive reset, FCCU or interrupt event by software.
ENABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	ENABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor disabled

Table continues on the next page...

Table 10-5. Voltage monitors configurability (continued)

POWERUP-PHASE0	PHASE1-2	DCF	REE[1]	RES[2]	FEE[3]	IE[4]	Monitor status
							during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] The LVD event is associated to an interrupt event and FCCU associated event is disabled.
ENABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	DISABLE	Monitor enabled during power-up, gating exit from power-up phase. Monitor disabled on phase0 exit. Monitor disabled during phase 3 on DCF record reading. An LVD event triggers an FCCU event. **After reset completion:**[5] The FCCU event associated to LVD event is disabled. After the software configuration, an LVD event will generate **no** event.
DISABLE	DISABLE	ENABLE	ENABLE	DEST	—	—	Monitor enabled during phase 3 on DCF record reading. Trigger reset event. Configuration locked to enable during run time.
DISABLE	DISABLE	ENABLE	ENABLE	FUNC	ENABLE	—	Monitor enabled during phase 3 on DCF record reading. Trigger destructive reset event. Configuration locked to enable during run time. **After reset completion:** LVD event demote from destructive reset to functional reset by software after completion of reset sequence. FCCU event is generated on functional reset.
DISABLE	DISABLE	ENABLE	ENABLE	FUNC	DISABLE	—	Monitor enabled during phase 3 on DCF record reading. Trigger destructive reset event. Configuration locked to enable during run time. **After reset completion:** Reset event demote from destructive reset to functional reset by software after completion of reset sequence. FCCU event is not generated.

Table continues on the next page...

Table 10-5. Voltage monitors configurability (continued)

POWERUP-PHASE0	PHASE1-2	DCF	REE[1]	RES[2]	FEE[3]	IE[4]	Monitor status
DISABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	ENABLE	Monitor disabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] The LVD event is also associated to an interrupt event.
DISABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	DISABLE	Monitor disabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] Configuration can be changed by software to any of following events: functional reset, destructive reset, FCCU or interrupt event by software.
DISABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	ENABLE	Monitor disabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] The LVD event is associated to an interrupt event whereas FCCU associated event is disabled.
DISABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	DISABLE	Monitor disabled during phase 3 on DCF record reading. An LVD event trigger an FCCU event. **After reset completion:**[5] The FCCU event associated to LVD event is disabled. After the software configuration, an LVD event will generate **no** event.

1. REE: Reset event enable
2. RES: Reset event select
3. FEE: FCCU event enable
4. IE: Interrupt enable
5. Configuration can be changed by software to any of following events: functional reset, destructive reset, FCCU or interrupt event by software

10.6 Power sequence

The following sections describe the power sequence and the relation between the different supplies during power-up and power-down.

The device is considered to be in a power sequence (or POWERUP state) when it is either not supplied or partially supplied. An internal power-on signal is used to identify POWERUP state. This signal is released high on exit of power sequence. The power-on signal is a combination of the LVDs monitoring the following supplies:

- VDD_LV
- VDD_HV_PMC
- VDD_HV_IO
- VDD_HV_FLA

The actual threshold use for each LVD is dependent on the configuration of the device. This is configurable by hardware (flash option bits content) or by software (LVD event configuration through register interface). Once power-on signal has been asserted, device configuration is reset to default power-up configuration.

10.6.1 Power-up sequence

In this section it is assumed that all supplies are low when entering the power-up sequence. Brown-out and power down sequences are managed in specific sections.

10.6.1.1 POWERUP state

At the beginning of power-up, the internal power-on signal remains low due to internal parasitics diodes. As soon as minimum threshold is reached on VDD_LV and VDD_HV supply, the internal power-on signal is forced low until power-up LVDs reach upper not trimmed threshold (refer to datasheet).

The different supplies can rise independently as long as the constraints described in the datasheet are met.

During power-up, all functional terminals are maintained into a known state as described in the following table.

Table 10-6. Functional terminal state during power-up and reset

Terminal type[1]	POWERUP[2] pad state	RESET pad state	DEFAULT pad state[3]	Comments
PORST	strong pull-down	weak pull-down	weak pull-down	Power-on reset pad
ESR0	strong pull-down	strong pull-down	weak pull-up[4]	Functional reset pad
ESR1	weak pull-up	weak pull-up	weak pull-up	Functional NMI/reset pad
TEST_MODE	weak pull-down	weak pull-down	weak pull-down	—
GPIO	weak pull-up	weak pull-up	weak pull-up	—
ANALOG	high impedance	high impedance	high impedance	—
ERROR	high impedance	high impedance	high impedance	During functional reset, pad state can be overridden by FCCU
JCOMP[5]	high impedance	weak pull-down	weak pull-down	—
TCK[5]	high impedance	weak pull-down	weak pull-down	—
TMS[5]	weak pull-up	weak pull-up	weak pull-up	—
TDI[5]	weak pull-up	weak pull-up	weak pull-up	—
TDO	high impedance	high impedance	high impedance	—

1. Refer to pinout information for terminal type
2. POWERUP state is maintained until supply cross the power-on reset threshold: V_{PORUP_LV} for LV supply, V_{PORUP_HV} for high voltage supply.
3. Before software configuration
4. ESR0 configuration move from strong pull-down to weak pull-up on completion of internal reset sequence (PHASE3[FUNC]).
5. JTAG interface implements build-in pull-up/pull-down configuration to ensure noise immunity. Worst case is boundary scan activation, requiring JCOMP to pulse high and following events to occur during JCOMP pulse high:
   - 23 TCK events corresponding to12 clock cycles.
   - 3 TMS events synchronous with TCK: sequence 0,1,1,0,0,0,0,0,0,0,0,1
   - 2 TDI events synchronous with TCK: sequence x,x,x,x,x,0,0,0,1,0,0,x

   TCK, TMS, TDI are implementing hysteresis requiring minimum variation

Digital reset is asserted, ensuring that all modules registers are reset to power-on value.

10.6.1.2 POWERUP exit

The POWERUP state can be exited when both VDD_LV and VDD_HV are above the internal LVD thresholds.

10.6.1.2.1 VDD_LV conditions and LVD trimming/enabling sequence

During POWERUP, only the following low voltage LVDs are enabled:

- LVD096_C

The VDD_LV conditions to exit POWERUP are the following:

- LVD096_C upper threshold is crossed

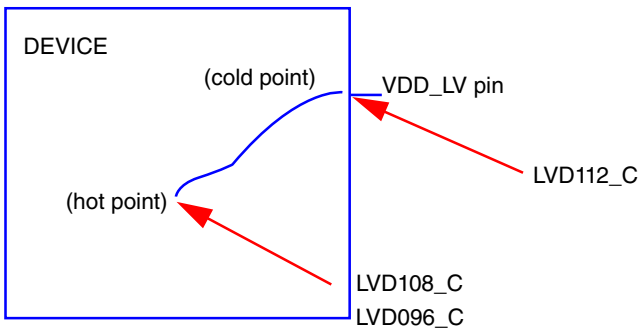


Figure 10-5. VDD_LV monitored voltages during power-up

After the POWERUP exit conditions (low voltage, high voltage conditions) have been verified, the internal power-on signal is released to all analog modules.

The internal RC oscillator module (IRCOSC) starts initialization and provides clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and the system counts a number of RC clocks. Once these clocks have completed and the LVD112_C has deasserted, the system will exit from PHASE0, LVD112_C is masked, and only LVD096_C remains active.

LVD096_C and LVD112_C share the same reference. LVD112_C – LVD096_C > 90 mV ± 5%. This difference provides margin with respect to maximum internal resistive drop (40 mV at I_{max}) and 50 mV hysteresis addressing external supply drop.

The device proceeds through the reset sequence through RGM phase, PHASE1[DEST], PHASE2[DEST] and PHASE3[DEST].

Voltage detector (LVD/HVD) modules are trimmed at the beginning of PHASE3[DEST]. Trimming of LVDs/HVDs are stored as internal DCF records. They are read by SSCM at low voltage, using the differential flash read accesses and applied to each LVDs/HVDs module. After a delay (including tLVDTRIM and other system operations) has elapsed, the PMC acknowledges that all LVDs/HVDs have been trimmed and supply is above threshold, the SSCM proceeds with the reset sequence, eventually running full speed accesses to the flash to read the option bits required to complete device configuration.

The configurable LVD/HVD modules can optionally be unmasked at the end of PHASE3[DEST]. Mask information is read as DCF record from the flash UTEST option bits.

When LVD108_C and LVD360_IM are masked by the application using the flash user option bits, the device should rely on PORST signal, being asserted by an external Voltage Detect circuit to detect a voltage failure during power-up. The device must wait for PORST to be released high before proceeding with the power-up sequence.This may increase the amount of time necessary to complete the reset sequence.

Figure 10-6 provides an example of VDD_LV power-up sequence from POWERUP to PHASE3[DEST].

Figure 10-7 illustrates threshold variation of VDD_LV LVD monitor during power-up sequence from POWERUP to PHASE3[DEST]. Internal LVD108_C LVD and internal HVD140_C HVD are *enabled* during PHASE3[DEST].

Figure 10-8 illustrates threshold variation of VDD_LV LVD monitor during power-up sequence from POWERUP to PHASE3[DEST]. Internal LVD108_C LVD and internal HVD140_C HVD are *disabled* during PHASE3[DEST]. The device relies on external voltage monitors.

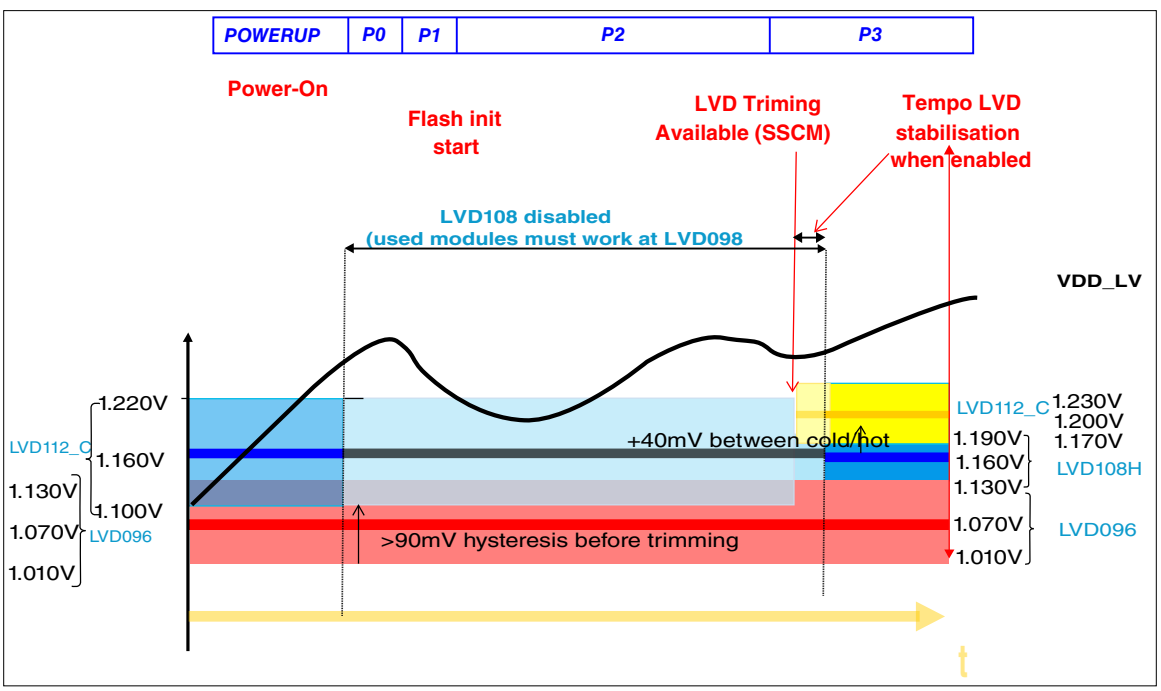


Figure 10-6. Example of VDD_LV power-up sequencing

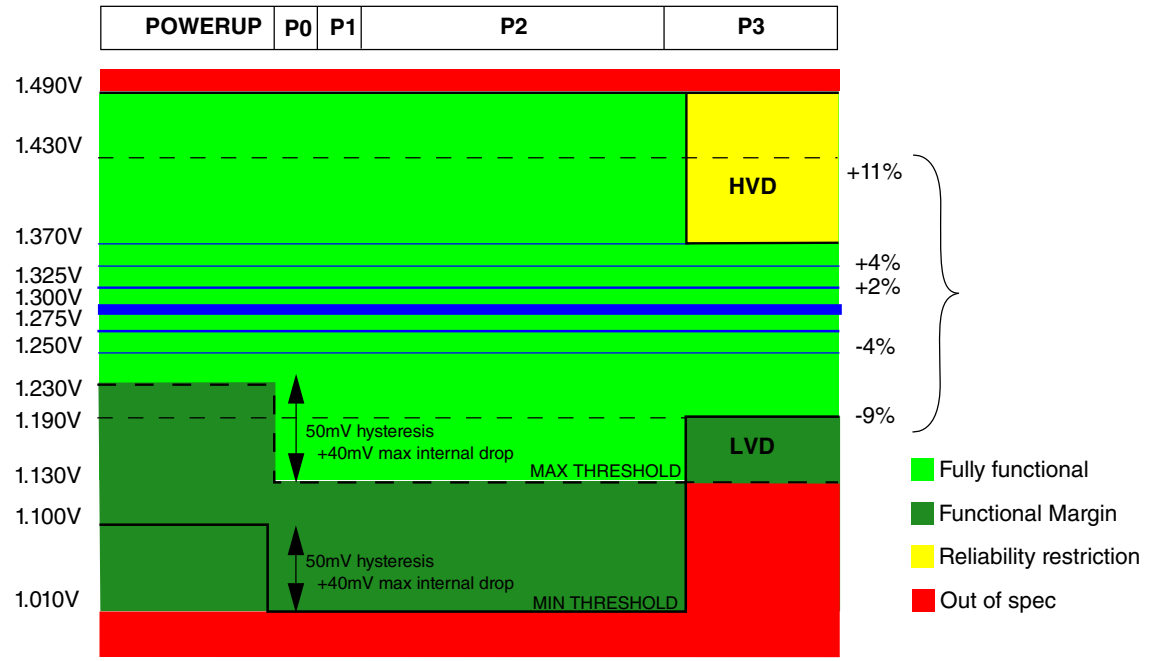


Figure 10-7. Threshold variation during power-up sequence: hot point LVD enabled

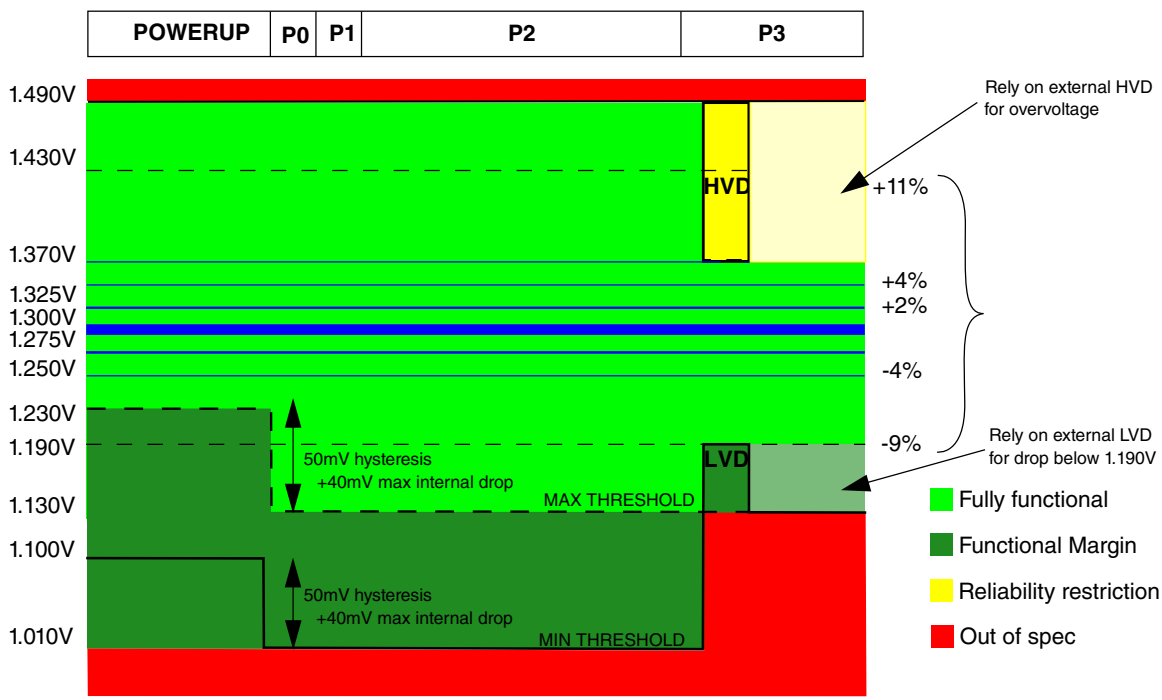


Figure 10-8. Threshold variation during power-up sequence: hot point LVD disabled

10.6.1.2.2 VDD_HV conditions and LVD trimming/enabling sequence

During POWERUP, the following high voltage LVDs are enabled:

- LVD295_A

- LVD295_F
- LVD270_xx

The VDD_HV conditions to exit POWERUP are the following:

- LVD295_A upper threshold is crossed.
- LVD295_F upper threshold is crossed.
- LVD270_xx upper threshold is crossed.

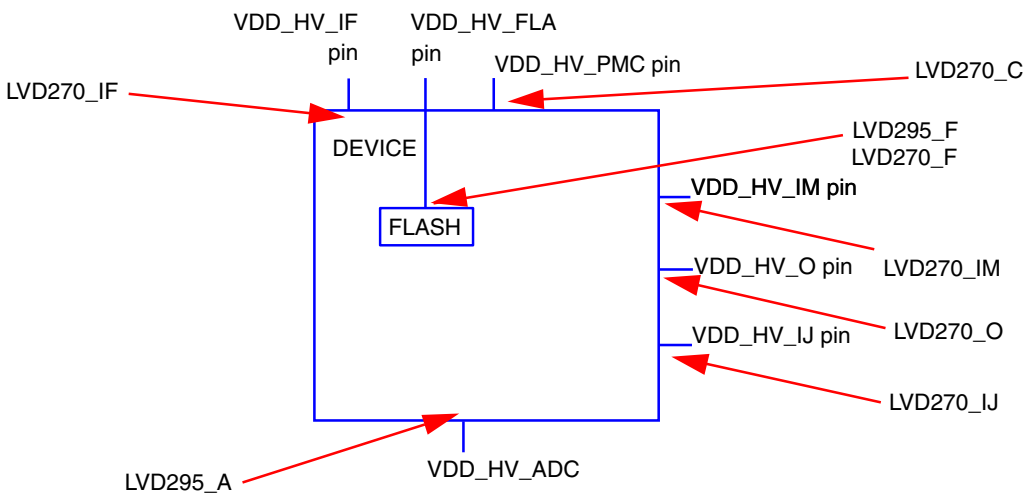


Figure 10-9. VDD_HV monitored voltages during power-up

After both LV and HV POWERUP exit conditions have been verified, the POR signal, ipg_por_b, is released to all analog modules.

The IRCOSC module starts initialization and provides the clock to the system after tRCSTARTUP. PMC digital interface reset is released after two RC clock cycles and the system counts a number of RC clocks. Once these clocks have completed and HVD600_C, LVD360_IM, and LVD400_IM have deasserted, the system will exit from PHASE0, and LVD295_A, LVD360_IM, and LVD400_IM are masked. All other HV LVDs remain active.

LVD360_IM and LVD400_IM share the same reference. LVD400_IM - LVD360_IM difference is implementing margin with respect to maximum internal resistive drop and hysteresis addressing external supply drop.

Device proceeds through the reset sequence through RGM phase, PHASE1[DEST], PHASE2[DEST], PHASE3[DEST]. PHASE2[DEST] will not be exited unless HVD600_C has deasserted.

LVD/HVD modules are trimmed at the beginning of PHASE3[DEST]. Trimming is done by SSCM at low voltage, using the differential flash read accesses. After LVD is completed, SSCM waits for PMC acknowledgement before proceeding with reset sequence.

Configurable LVD/HVD modules are optionally enabled at the end of PHASE3[DEST].

Figure 10-10 provides the threshold variation of VDD_HV LVDs monitor during power-up sequence from POWERUP to PHASE3[DEST]. Internal LVD360_IM, LVD270_XX, LVD400_A, LVD400_IM LVDs and internal HVD600_IM, HVD600_C, HVD360_IM HVDs are *enabled* by default during PHASE3[DEST] although this can be controlled via the LVD/HVD enable DCF load.

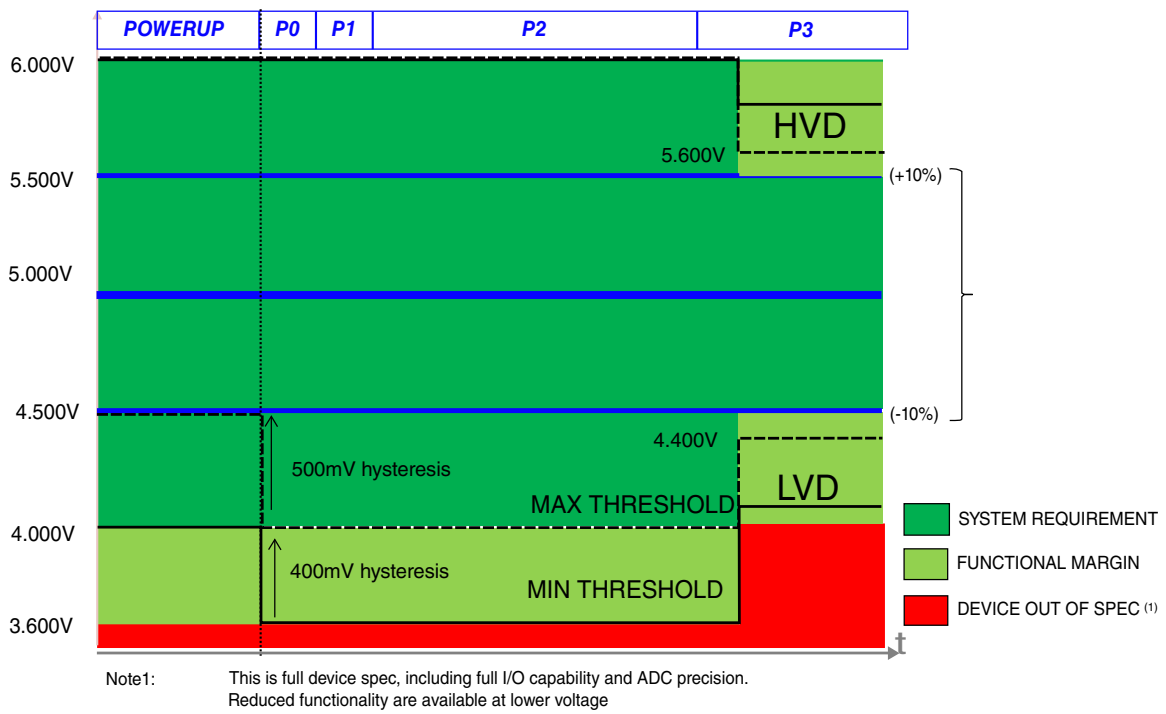


Figure 10-10. Threshold variation during power-up sequence — 2.7V–5.5V hot point LVD disabled, 4.5V–5.5V range

The device needs to fulfill all conditions described on Power-up sequence to recover.

10.6.2 Power-down sequence

In case the threshold of the higher ranges LVDs that are configured to generate destructive reset is crossed but neither LVD096 nor LVD270 threshold is crossed, the device enters PHASE0 phase.

Power-down sequence is actually entered as soon as the threshold of one of the LVD096 or LVD270 is crossed. The device enters the POWERUP state. The internal power-on signal ipg_por_b is asserted.

On ipg_por_b assertion, all analog modules reach their safe state.

Device supplies may then proceed to drop down to ground either through device leakage or external pull-down. During power-down, all supplies should comply with supply constraints.

In the case of an intentional power down of the Low Voltage supply (1.2V), the Auxillary Regulator should be disabled by writing a 1 to the bit VREG1P2_CTRL[VREG1P2_DIS].

10.6.3 Brown-out management

During brown-out, the device re-enters the POWERUP phase as soon as the threshold of either LVD096 or LVD270 is crossed. The internal power-on signal ipg_por_b is asserted.

Note

The device will correctly start independently from any residual voltage. Below LVD270 threshold the device is under POWERUP state. Above the LVD270 threshold, the device is into a RESET state as defined by Voltage monitor or external PORST pin or active state when all have been released.

WARNING

This mechanism is ensured when LVD270 are not disable by the user. In case LVD270 are disabled, reset state must be ensure by assertion of external PORST pin from the user. When supply goes below the threshold of the POR240 (internal power-on state detector), device will be forced into reset and LVD270 is automatically re-enabled.

On internal power-on signal assertion, all analog modules reach their safe state.

10.6.4 Low voltage requirement during crank

The device can continue operation to the minimum input voltage during crank.

In order to proceed with execution during cranking and prevent device reset, it is important to correctly configure the high voltage LVDs.

Internal LVDs that monitor the supply ensure that the flash content is protected.

Chapter 11
Security

11.1 Introduction

All MPC57xx family devices have a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. Some security features are available in all device configurations while other more advanced security features are not available in all device configurations. These more advanced features require more than simply being turned on or off; they require significant involvement for implementation. Consequently, they provide an important opportunity to define the security level of the device.

11.2 Basic security

All MPC5777M devices have the following basic set of security features.

- Device censorship based on the life cycle model with code and data access progressively more restricted as device matures through defined life cycle steps
- Memory security features:
    - NVM censorship support, password protection, one-time-programmable (OTP) flash memory areas, Flash erase counter and tamper detection
    - SRAM and caches initialized to a constant value after reset (power-on reset, when in a test mode, when not booting from internal flash memory); MBIST functionality used in case of power-on and destructive resets to initialize RAMs selected via configuration in UTEST flash; BAF software routines to initialize RAMs when not booting from internal flash memory
- Monitoring of operating conditions

- Unique ID for each device: each MPC5777M device has a unique identification number stored in an OTP flash memory area which can be read by any of the cores on the device
- Secure watchdog timer
- Basic debugger restrictions (on/off via censorship mode)

11.3 Advanced security

Depending on the device configuration, the following features are available:

- Secure debugger interface
- Boot modes:
    - Trusted/secure boot support

11.4 Detailed security information

Details of most of the MPC5777M security features are published in a separate *MPC5777M Microcontroller Security Reference Manual*, which is only available to qualified customers.

Chapter 12
Calibration and Debug

12.1 Introduction

This chapter discusses the device-specific information for the debug and calibration modules. This device implements a high-speed serial Nexus trace auxiliary port. The Aurora interface allows the Nexus protocol information to be transmitted serially at high speed over four Aurora lanes. The Aurora protocol handles the encoding of the data and striping the information across the four Aurora lanes available on the device. In addition, the LFAST module, a high speed calibration interface, is supported for run control and memory access. The LFAST signals are alternate functions on the device's JTAG pins. The device starts up in JTAG mode and the tool can request that the JTAG interface be converted to the LFAST high-speed interface. All standards used have freely available specifications for tool developers.

The debug and calibration features for this device are provided in two separate dies: the Production die and the Buddy die, or Buddy Device (BD). The Production die is packaged alone in the Production Device (PD) package. The PD is also assembled into an Emulation and Debug Device (ED) package that consists of the Production die along with the Buddy die. The Buddy device supports additional debug capability. The Nexus Aurora interface is only available on the ED. The PD only supports trace to the internal trace/calibration memory.

The Event Out ($\overline{\text{EVTO}}$[0,1]) signals are connected to the external TGOUT[0,1] device pins and the Event In ($\overline{\text{EVTI}}$) signals are connected to the external TGIN[0,1] device pins.

12.2 Core debug support

Internal debug support in the e200z710n3 and e200z425n3 cores allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. The cores have the capability of exiting reset halted in debug

mode, such that no code runs. This allows debuggers to perform operations such as configuration of calibration remap or edit of overlay memory prior to user code execution.

For details, see e200z425Bn3 Core Debug Support and e200z710n3 Core Debug Support chapters.

12.3 Run control and memory access

Run control is any operation required to control device operation. Run control includes starting and stopping the processor's execution of code, as well as accessing memory while the device is stopped to download code. The Debug and Calibration Interface (DCI) provides the central run control including cross-triggering of breakpoint conditions. This DCI provides support for the IEEE 1149.1 JTAG standard with the JTAG Controller (JTAGC) as well as the IEEE 1149.7 standard with the Compact JTAG (CJTAG) module. The JTAG interface supports both Boundary Scan and Debug modes. It is based on the IEEE 1149.1 and IEEE 1149.7 standards. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface. The JTAGC can either be fed directly from the JTAG pins or the CJTAG module, which supports the IEEE 1149.7 standard.

The Sequence Processing Unit (SPU) provides cross-triggering of events from the device. The SPU can access some of the DCU run control features since it can control the $\overline{\text{EVTO}}$ pins and monitor the $\overline{\text{EVTI}}$ pins. Furthermore, the SPU can use system triggers to cause many different types of actions, including triggering a breakpoint cross-trigger (managed by the DCI).

Table 12-1. Reference links to related information

Related module	Reference
Production Device Debug and Calibration Interface (DCI)	Debug and Calibration Interface (DCI)
JTAG Controller (JTAGC)	JTAG Controller (JTAGC)
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)
JTAG Data Communication (JDC)	JTAG Data Communication (JDC)
Sequence Processing Unit (SPU)	Sequence Processing Unit (SPU)

12.3.1 Debug and Calibration Interface (DCI)

The Debug and Calibration Interface (DCI) module provides debug and calibration features. The DCI module includes the device JTAG Controller, the IEEE 1149.7 interface. The DCI provides the following features:

- Debug mode enable control for connected tools or software
- Port-sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Simultaneous operation for debug and calibration features

12.3.1.1 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG Controller.

12.3.1.1.1 JTAGC ACCESS_AUX block instructions

The ACCESS_AUX instructions are described in these tables.

Table 12-2. Production die ACCESS_AUX 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
Reserved	100000	Reserved
ACCESS_AUX_NR_PD	100001	Enables access to the NAR TAP controller on PD
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved

Table continues on the next page...

Table 12-2. Production die ACCESS_AUX 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Factory test reserved	100101	Intended for factory test only
ACCESS_AUX_JDC	100110	Enables access to the JDC TAP controller, to access the HSM challenge and response registers
ACCESS_AUX_HSM	100111	Enables access to the HSM core TAP controller
ACCESS_AUX_CORE_0	101000	Enabled access to E200 Core 0
ACCESS_AUX_CORE_1	101001	Enables access to E200 Core 1
ACCESS_AUX_CORE_2	101010	Enables access to E200 Core 2
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
ACCESS_AUX_GTM	110011	Enables access to GTM TAP controller
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL[1]	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
Reserved	111110	Reserved

1. The use of this instruction is limited to the synchronous exit of debug mode via the execution of the GO+EXIT OnCE command. The execution of other instructions or commands is not supported

Table 12-3. Buddy die ACCESS_AUX 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
ACCESS_AUX_NR_BD	100000	Enables access to the NAR TAP controller on BD
Reserved	100001	Reserved
ACCESS_AUX_NAL	100010	Enables access to the NAL TAP controller
ACCESS_AUX_RWA_BD	100011	Enables access to the RWA TAP controller
Reserved	100100	Reserved

Table continues on the next page...

Table 12-3. Buddy die ACCESS_AUX 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Factory test reserved	100101	Intended for factory test only
Reserved	100110	Reserved
Reserved	100111	Reserved
Reserved	101000	Reserved
Reserved	101001	Reserved
Reserved	101010	Reserved
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
Reserved	110100	Reserved
Reserved	110101	Reserved
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
ACCESS_AUX_DWPU	111001	Enables access to DWPU controller
Reserved	111010	Reserved
Reserved	111011	Reserved
Reserved	111100	Reserved
Reserved	111101	Reserved
ACCESS_AUX_DCI_PD	111110	Enables access to pass control from BD to PD

12.3.1.2 Compact JTAG (CJTAG)

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities.

12.3.2 JTAG Data Communication (JDC)

The JTAG Data Communication (JDC) module allows both JTAG and software access to two registers. These two registers are the challenge/response password authorization registers used by the Hardware Security Module (HSM).

12.3.3 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, by using internal device signals as trigger sources. Performance monitor functions are available at both the system and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System-level performance monitor functions are integrated into the SPU complex trigger logic, thus allowing the counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in Debug mode. The SPU collects these triggers and uses them as conditions for programmable sequences of states and resultant actions. The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, it achieves complex logic-analyzer-like behavior, providing vital real-time visibility and the ability to debug system activities.

12.4 Calibration interface

This device includes an alternate calibration interface for high-speed run control and advanced software control. This calibration interface is based on the LFAST interface. The LFAST based calibration has the ability to perform all of the functions that JTAG supports, only at a faster speed. The MCU implements the slave module, and the pins for the LFAST interface are multiplexed onto the same pins as the JTAG pins. Initially, the device powers up with the JTAG module in control of the pins. Under tool commands, the LFAST interface can be enabled and the pins become LFAST signals. Once the LFAST interface has been activated, all run control and other JTAG operations are converted in the MCU from LFAST messages into JTAG control operations of the JTAG Controller. The interface is supported entirely in hardware for both LFAST and JTAG modes, with no software or system overhead required to support interface traffic.

Table 12-4. Reference links to related information

Related module	Reference
JTAG Master (JTAGM)	JTAG Master (JTAGM)
LFAST Module—JTAGM	LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug
Development Tool Semaphore (DTS)	Development Tool Semaphore (DTS)

12.4.1 JTAG Master (JTAGM)

The JTAG Master (JTAGM) acts as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or through software.

The data transferred to this module is transformed to produce TCK, TMS, TDI, and TRST outputs and to accept TDO inputs. The module allows these five signals to connect to the JTAGC as if the JTAG data were coming from an external tool. The JTAGM generates all required JTAG scan chains to allow software and high-speed serial communication access to all JTAG-mapped resources.

12.4.2 Debug LFAST

The high-speed calibration interface provides up to a 320 Mbit/s interface between a calibration tool and the device. The calibration interface uses the LFAST as the physical layer between the tool and the JTAGM for access to debug resources. In addition, an LFAST switch allows direct access to read and write memory. The calibration debug LFAST module is one of two instantiations of the LFAST module on the device. The calibration LFAST is configured for slave-only operation and is meant to be used as a calibration interface to provide higher (than JTAG) speed debug and calibration operations. The calibration LFAST interface supports full duplex operation, where the available bandwidths for upstream and downstream traffic are independent.

12.4.3 Development Trigger Semaphore (DTS)

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways that this module can be used, including as a component of an external real-time data acquisition system.

Note

When used as a component of a triggered data-acquisition system, Nexus read/write access is via the JTAG interface of the Nexus debug port and is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus auxiliary port.

12.5 Debug over CAN

As well as supporting debug via JTAG and LFAST interfaces, the device supports the use of a CAN interface for debug. This allows debug of application hardware where access to the dedicated JTAG/LFAST interface signals is not easily available. The debug over CAN mechanism is intended to provide basic debug functionality with some reduction in bandwidth and features compared to use of the dedicated JTAG or LFAST interfaces.

Use of the CAN interface for debug purposes requires the use of some M_CAN, eDMA and JTAGM resources. Some initialization of these resources via software is required, but once this initialization has completed, debug over CAN is possible without any further software overhead.

As the debug over CAN scheme generates internal JTAG messages based on received CAN data, all JTAG clients and included debug resources are accessible. Basic device trace capability is also possible by configuring the device trace hardware to stream to a device overlay/trace RAM, which can be read later using debug over CAN.

The debug over CAN scheme supports the following features:

- Supports operation through M_CAN_0 or MCAN_1 interfaces
  - Makes use of M_CAN debug enhancements
  - Debug message selected by filter configuration SFEC/EFEC=111
- Allows debug of hardware where JTAG access is not available
- Debug traffic carried over application CAN bus
- Debug traffic on CAN coexists with application traffic
- No software overhead after initialization
- Flexible selection of CAN identifiers for debug use
- Access to all JTAG debug facilities, includes CPU run control

12.6 Nexus Trace Aurora interface

The Aurora interface allows Nexus protocol information to be transmitted serially at high speed over multiple Aurora lanes to provide advanced trace information from the device. The Aurora protocol handles the encoding of the data and the striping of information across the lanes. The Aurora interface is only available on the Emulation and Debug Device. Trace information is transferred to the Buddy Device for transmission via the Aurora interface.

The Aurora information can be used to reconstruct events or operations that occurred inside the microcontroller. Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the MCU. Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client; generally, it can be individually enabled (both the type of message and the client). Table 12-5 shows a few examples of Nexus messages. All messages can be enabled to include a timestamp.

Table 12-5. Nexus message types

Message Type	Description
Program trace	Nexus program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership trace	Ownership trace provides information on process identification changes.
Watchpoint trace	Watchpoint messages are generated any time a watchpoint match occurs in the program or data flow, or as a result of SPU actions. These can be programmed for many types of events within the MCU.
Device identification	The device identification message allows information about the MCU to be transmitted upon startup to allow tools to identify the target system MCU type.
Debug status	Debug status messages provide additional information that may be needed to reconstruct software execution, such as whether the device has entered Debug or Low-power mode.
Data acquisition	Data acquisition messages are an optional feature that allows software control of information to be transmitted.
Error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

The Nexus trace information can be output to internal memory, or the parallel information can be serialized for transmission over a serial interface through the Nexus Aurora interface. Nexus trace information can also be routed through the Data Write Processing Unit (DWPU) as described in Data trace filtering.

The Nexus blocks do not have memory-mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in the JTAGC chapter. Next, after the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The

enable instructions for the different Nexus clients are shown in the following table. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 12-6. Nexus client JTAG enable instructions

Module	Enable instruction	Opcode
Production Die		
Cores	CORE_ENABLE	0x7C (10 bits)
NXMC	NEXUS_ENABLE	0x0 (4 bits)
GTM		
NAR		
SPU		
Buddy Die		
RWA	CORE_ENABLE	0x7C (10 bits)
NAR	NEXUS_ENABLE	0x0 (4 bits)
NAL		
DWPU		

All of the internal JTAG modules are accessed through a single JTAG connection to the device. In the case of the ED, all JTAG communication is performed through the JTAG interface located in the BD. In the production package configuration, access is performed through the JTAG interface located in the PD.

With the hierarchical JTAG configuration, the BD JTAGC is considered the master JTAG TAP controller. The PD JTAGC TAP controller is selected via an AUX_ACCESS instruction. Other TAP controllers residing on the BD are also selected via AUX_ACCESS instructions (NAR, RWA, BD_DCI). Once control is transferred from the BD JTAGC to the PD JTAGC via the AUX_ACCESS instruction, all other PD TAP controllers are then selected via AUX_ACCESS instructions executed by the PD JTAGC. This is also referenced as TAP-sharing and is shown in the following figure.

NOTE

On the Emulation device, when the PAUSE-DR/UPDATE-DR state is executed to pass control of the JTAG TAP controller, control is always passed to the top level JTAG TAP controller, in other words, the DCI of the BD. For this reason, it is recommended that all JTAG operations, except when the BD clients need to be accessed, that all PD JTAG access be preceded with ACCESS_PD_JTAGC command. If the device is not an ED, the access is ignored.

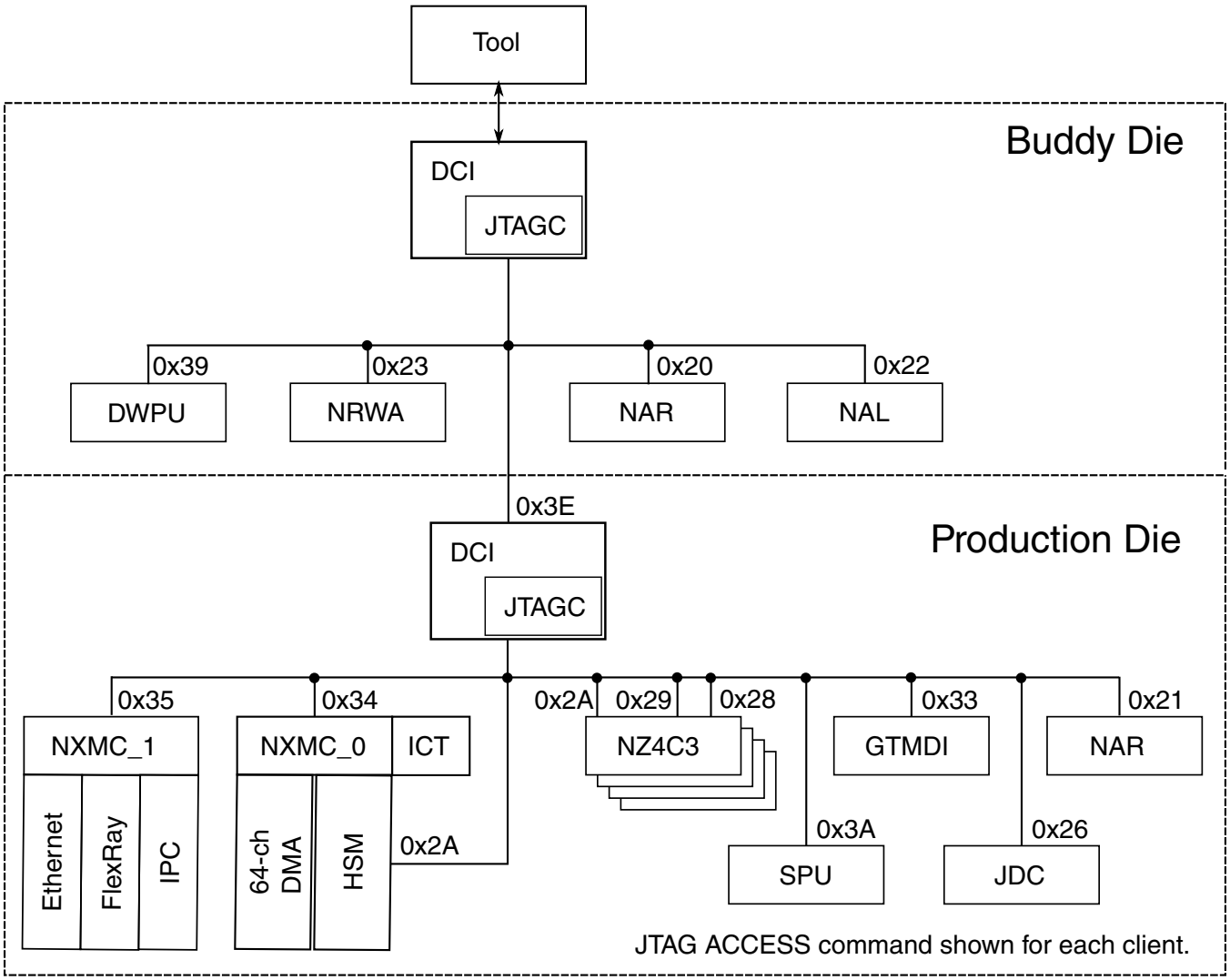


Figure 12-1. JTAG hierarchal view

Table 12-7. Reference links to related information

Related module	Reference
Production Device Nexus Aurora Router (NAR)	Production Device Nexus Aurora Router (NAR)

12.6.1 Nexus Aurora overview

The IEEE-ISTO 5001-2011 standard adds a new type of auxiliary trace port that supports a higher bandwidth. This new interface type uses the Aurora protocol to reformat the parallel Nexus messages into one or more output serial lanes of data, where a lane is defined as a 2-pin Low Voltage Differential Signals (LVDS) interface. The protocol handles striping the data onto the multiple lanes. Each lane uses an 8b/10b encoded LVDS driver. The encoding allows the receiver to recover clocking information.

The Aurora physical interface is based on the XAUI (10 Gigabit Attachment Unit Interface) standard used in communication protocols, which covers the jitter, differential skew, inter-lane skew, and bit error rate requirements.

The Aurora protocol covers lane initialization and channel bonding for both simplex and duplex implementations. The current implementation uses a simplex output-only connection from the MCU to the external tool.

The Aurora interface connects Nexus parallel trace output to an Aurora Protocol Engine (APE), the Aurora Lane Control (ALC), and the Nexus Aurora Physical interface (NAP). Figure 12-2 shows the Aurora interface block diagram, where the number of lanes, n, is 4.

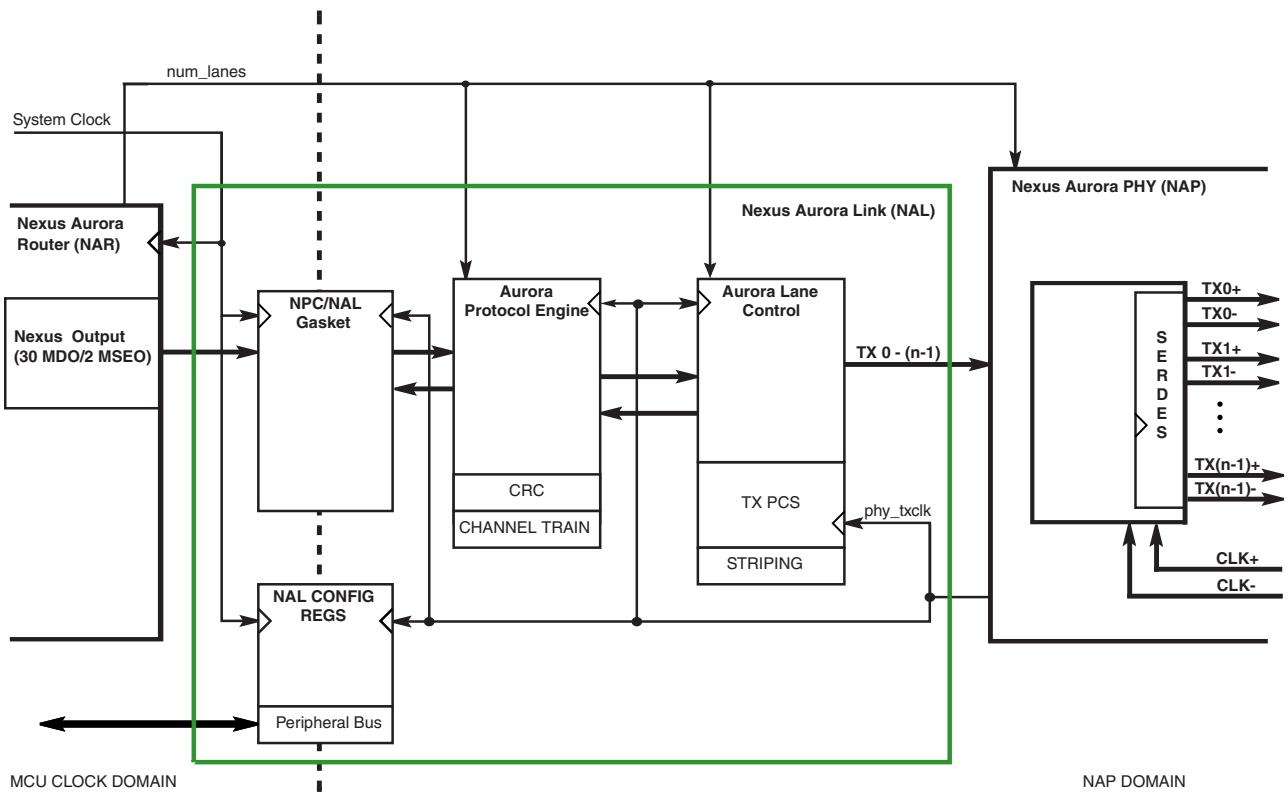


Figure 12-2. Aurora interface block diagram

12.6.2 Nexus Aurora Router (NAR)

The Nexus Aurora Router (NAR) manages the Nexus trace information from each of the Nexus clients. The NAR takes the parallel trace information from each of the connected clients, buffers it to avoid overruns, and routes it to either the internal overlay RAM or the Buddy Device, where it is stored in on-chip SRAM or transmitted out of the device through the Aurora interface.

16 KB of trace SRAM is available on the PD (located in the flash module), and 1 MB of trace SRAM is available in the ED. (Some of this SRAM may be used for calibration overlay content as well as trace content.) Similarly, a smaller NAR output queue is located on the PD (2 KB), and a much larger NAR output queue is located in the ED (8 KB). The physical Aurora pins are only available on the ED. Figure 12-3 shows the NAR block diagram.

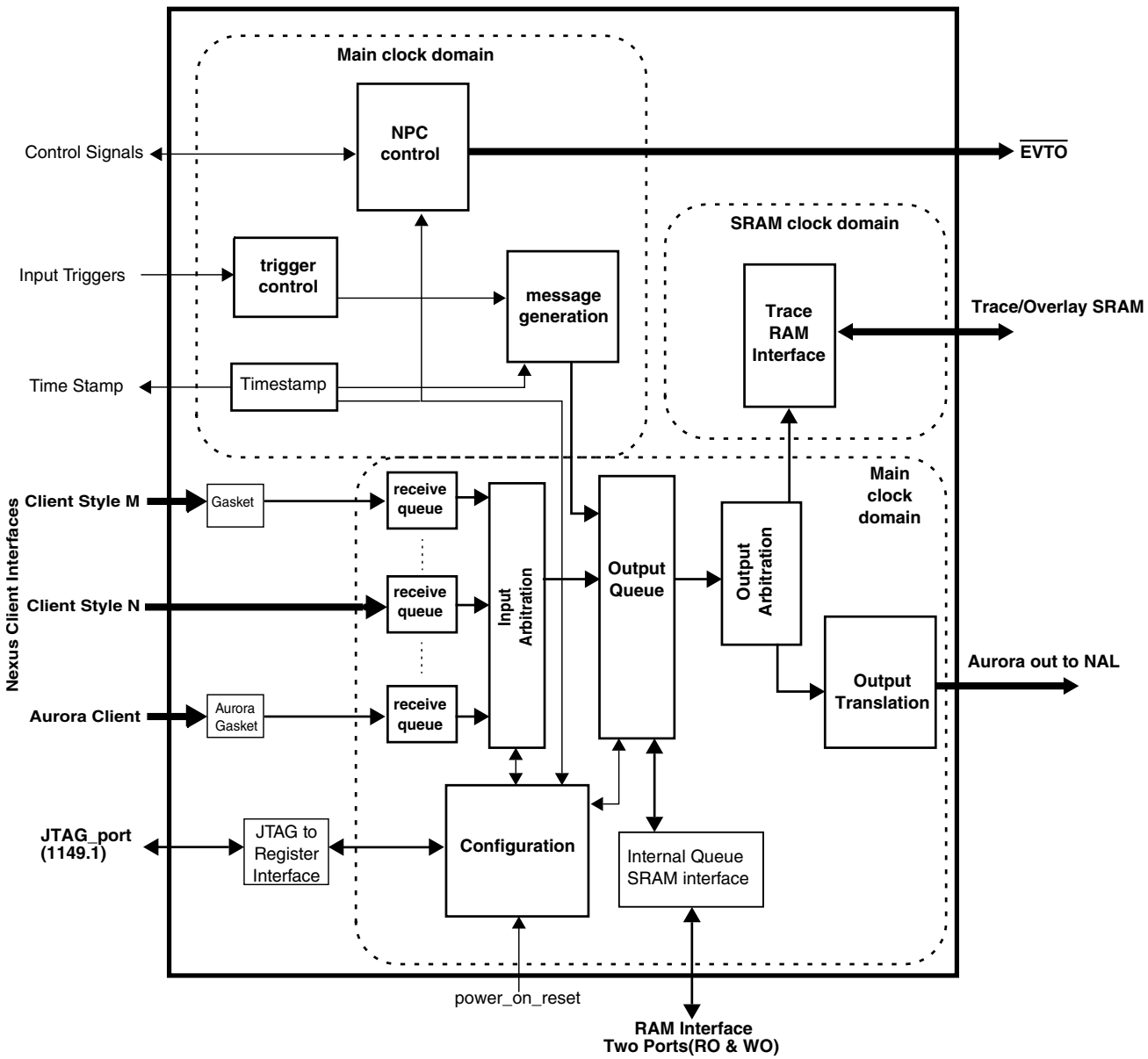


Figure 12-3. NAR block diagram

12.6.2.1 NAR clients

The PD NAR module has seven clients, with client index 1–7. Clients 8–16 are not connected. The client connections and the corresponding NAR client index are shown in the following table. The NAR client index is used in the NAR_STCR, NAR_CSSR, and NAR_CDR registers. The source trace ID is used in the NAR_SFR.

Table 12-8. PD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
—	NAR	0b1111 (0xF)	—
1	GTM	0b0111 (0x7)	3
2	HSM	0b0011 (0x3)	1
3	Core 0	0b0000 (0x0)	1
4	NXMC Client 0 Concentrator	0b1100 (0xC)	1
5	Core 1	0b0001 (0x1)	1
6	NXMC Client 1 Concentrator	0b1101 (0xD)	1
7	Core 2	0b0010 (0x2)	1
8	—	—	—
9	—	—	—
10	—	—	—
11	—	—	—
12	—	—	—
13	—	—	—
14	—	—	—
15	—	—	—
16	—	—	—

The BD NAR module has one client and the client connections and the corresponding NAR client index are shown in the following table.

Table 12-9. BD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
1	PD NAR	0b1110 (0xE)	2

The NAR is limited writing trace information to the overlay RAM. The PD NAR only has access to the 16 KB internal overlay RAM. Therefore, the PD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0D000000–0x0D003FFF. The BD NAR also has access to 1 MB of extended overlay RAM. Therefore, the BD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0C000000–0x0C1FFFFF.

The NUM_LANES field of the NAR_NAPCR is described in the following table.

Table 12-10. NAR_NAPCR field descriptions

Field	Description	
19–16	TX Lane Configuration. Number of enabled TX lanes.	
NUM_LANE	0000	Disable NAP and NAL
	0001	Reserved
	0010	Enable 2 lanes (Lane 0–1)
	0011	Reserved
	0100	Enable 4 lanes (Lanes 0–3)
	0101	Reserved
	0110	Reserved
	0111	Reserved
	1000	Reserved
	1001–1111	Reserved

12.6.3 Nexus Aurora Link (NAL)

The Nexus Aurora Link (NAL) is only included in the Emulation and Debug Device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer serializer and transmitter (no receiver). It includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol.

Debug data in the Nexus format from multiple clients is packetized into a 32-bit format by the clients and then collected by the NAR. The NAL encodes the message data (8b/10b), stripes it across the lanes, and inserts Aurora control characters (such as start of message, end of message, idle, clock compensation characters, and so on) around the payload. The NAL PCS drives the Nexus Aurora Physical (NAP) interface.

12.6.4 Nexus Aurora PHY (NAP)

The Nexus Aurora PHY (NAP) is only included in the Emulation and Debug Device. The NAP, in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. The NAP receives 8b/10b encoded data from the ALC block in the NAL and transmits this data onto a given PHY lane's LVDS pads. The NAL performs lane striping with settings in the NAL and NAR specifying lane configuration. Each NAP lane consists of a Data Symbol Character Accumulator that registers the 10-bit character for that lane from the ALC on the rising

edge of the NAL clock. A serializer takes the 10-bit character and serializes it into the individual transmit data bits at the PHY clock rate to be transmitted via the LVDS output drivers to the external debugger receive channel.

Note

The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive data side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called "lane alignment" in Aurora terminology), 10b/8b decoding, decode and disparity error tracking, and CRC checking in the received data stream.

Figure 12-4 shows a block diagram of the Nexus Aurora Physical (NAP) interface and the LVDS output drivers, where the number of lanes, n, is 4.

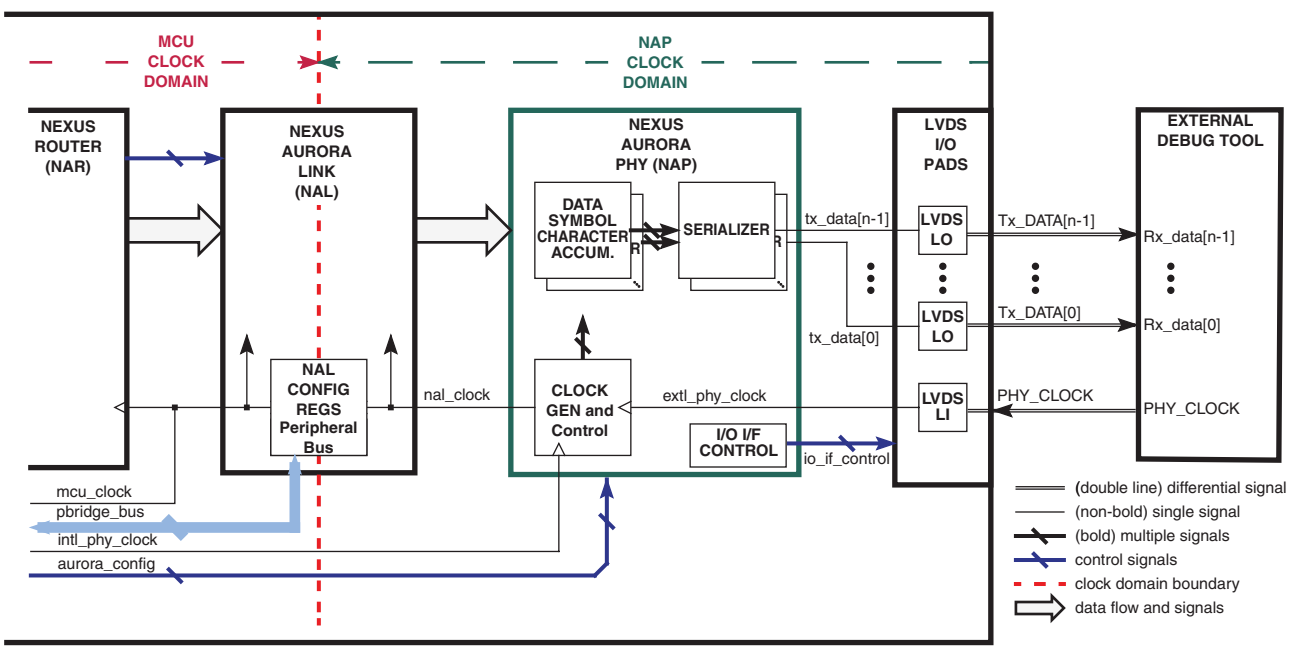


Figure 12-4. NAP block diagram

12.7 Nexus clients

Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the device.

Table 12-11. Reference links to related information

Related module	Reference
e200z4 Nexus Support	Nexus Module
GTM Development Interface (GTMDI)	GTM Development Interface (GTMDI
Nexus Crossbar Multi-Master Client (NXMC)	Nexus Crossbar Multi-Master Client (NXMC)

12.7.1 e200z Nexus 3

The e200z710n3 and e200z425n3 Nexus 3 modules provide real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering and Processor Overrun Control.

12.7.2 GTM Development Interface (GTMDI)

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, Input Channel Filters trace, Output Timer channels, DPLL and Sensor Pattern Evaluation (SPE) module submodules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to implement complex debug features. Trace capability is also provided along with a global timestamp that allows real-time trace for several GTM submodule events. The GTMDI implements a Nexus client standard interface and a JTAG standard port controller to provide communication capability via the JTAG port and the on-chip NAR.

12.7.3 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard.

12.7.3.1 NXMC block connections

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Two NXMCs are implemented: one for the fast masters and one for the slow masters. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In addition to XBAR bus master trace capability, the NXMC also allows for in-circuit trace capability that can be used to trace the Sequence Processing Unit (SPU) counter values. The two external hardware watchpoint triggers of each NXMC are used to generate timestamps (four total—two from each NXMC) for events from the SPU.

The following figure shows a block diagram illustrating the NXMC integration.

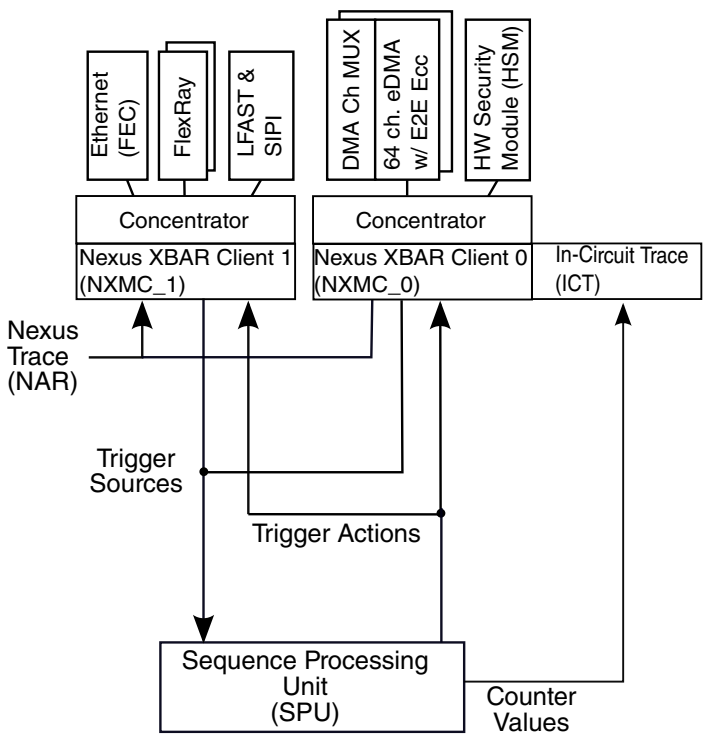


Figure 12-5. NXMC integration block diagram

The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 12-12. NXMC source and master client IDs

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
In-Circuit Trace	SPU	0b1011	0b0000
XBAR Client 0 Concentrator (fast masters)	Hardware Security Module (HSM)	0b1100	0b0000
	eDMA Controller 0		0b0001
	eDMA Controller 1		0b0010
	Reserved		0b0011 –0b0111
XBAR Client 1 Concentrator (slow masters)	System Bus	0b1101	0b1000
	FlexRay 0		0b1001
	Ethernet		0b1010
	FlexRay 1		0b1011
	Reserved		0b1100–0b1110
Data Write Processor Unit (DWPU)	e200zx core	—[1]	0b1111

1. The Nexus trace Source ID is the same as the original core SRC.

12.8 Data trace filtering

The Data Write Processing Unit (DWPU) supports data trace filtering as well as out of range parameter identification.

Table 12-13. Reference links to related information

Related module	Reference
Data Write Processing Unit (DWPU)	Data Write Processing Unit (DWPU)

DWPU operates in conjunction with the BD NAR and a small RAM array. DWPU receives its input trace stream from the BD NAR Nexus Out Link. The trace stream is initially decompressed (which includes conversion of relative address and data values to absolute values) to ease subsequent filtering and comparison operations. The decompression process tracks multiple contexts, with individual context tracking required for each supported Nexus client.

The processed stream then passes through an address compare block, that generates tag RAM addresses corresponding to a portion of each data write address within the trace stream. Another portion of the data write addresses is used to select a particular bit from the data word returned by the tag RAM, so that each tag RAM bit corresponds to a unique 64-bit data write address. The value of the selected bit determines whether the

data write trace occurrence is passed out of the DWPU back to the BD NAR, or is removed from the trace stream. DWPU returns the data trace filtered and re-formatted stream back to a secondary Nexus client input port on the BD NAR.

Chapter 13
Core Complex Overview

13.1 Introduction

The MPC5777M has four separate cores that perform various computational and control functions. The main computational shell consists of Main Core_0 and Main Core_1, both of which use an e200z710n3 core. These two cores are used for general computational functions. A third core, referred to as Checker Core_0s, uses an e200z709 core, which is a true subset of the e200z710n3 core. When enabled, Checker Core_0s operates in Lock Step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to ensure that Main Core_0 is executing correctly. There is no Lock Step function associated with Main Core_1. The fourth processor, Peripheral Core_2, employs an e200z425Bn3 core and is generally used to control various I/O modules.

NOTE

The processor ID for each CPU core is contained in its Processor ID Register (PIR).

This chapter provides overviews of the e200z710n3 core, the e200z709 core, and the e200z425Bn3 core.

13.2 Overview of the e200z710n3, e200z709, and e200z425Bn3 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power Architecture. The e200z710n3, e200z709, and e200z425Bn3 cores are very similar. The register set for all three of these cores is identical. The e200z425Bn3 core differs in hardware from the e200z710n3 core in several ways. The e200z425Bn3 core and e200z710n3 core have different amounts of Instruction and Data Cache. The e200z425Bn3 has no Data Cache. Also, the e200z425Bn3 incorporates a Lightweight

Signal Processing Extension (LSP) APU to provide support for real-time SIMD fixed point embedded numeric operations. The e200z709 core has no Instruction Cache, Data Cache, I-memory, or D-memory.

13.3 Features

The following is a list of some of the key features of the e200z710n3, e200z709, and e200z425Bn3 processor cores:

- Two integer execution units
  - Branch control unit
  - Instruction fetch unit
  - Load/store unit
  - Multi-ported register file
    - Capable of sustaining six read and three write operations per clock
    - Most integer instructions execute in a single clock cycle.
  - Branch target prefetching is performed by the branch unit
    - Allows single-cycle branches in many cases.
- Dual-issue (e200z425Bn3), 32-bit *PowerISA 2.06*-VLE compliant CPU
  - VLE ISA for improved code density
  - *PowerISA 2.06* fixed length 32 bit instruction set is not directly supported
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using BTB
- Load/store unit
  - 2 cycle load latency
  - Fully pipelined
  - Misaligned access support
- 32-bit General-Purpose Register (GPR) file
- Dual AHB 2.v6 64-bit system buses

- Local Instruction and Data Memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z710n3 core-16 KB IMEM, 64 KB DMEM
  - e200z425Bn3 core-16 KB IMEM, 64 KB DMEM
  - e200z709 core does not contain either Instruction or Data Memories
- Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions
- 2-Way Set Associative Harvard Instruction and Data Cache
  - e200z710n3 core-16 KB ICache, 4 KB DCache
  - e200z425Bn3 core-8 KB ICache, no DCache
  - e200z709 core does not contain either Instruction or Data Cache
- Embedded Floating-point (EFPU2) APU
  - supports real-time single-precision embedded numeric operations
  - uses general-purpose registers
- Lightweight Signal Processing (LSP) APU (e200z425Bn3 core only) supporting fixed-point multiply/accumulate and integer/fixed-point operations using the 32-bit General-Purpose Register file.
- Performance Monitor APU supporting execution profiling
- Nexus Class 3+ Real-time Development Unit for e200z710n3 and e200z425Bn3
- Power management
  - Low power design with extensive clock gating
  - Power saving modes: DOZE, NAP, SLEEP, WAIT
  - Dynamic power management of execution units, caches, and local memories
- Testability
  - Synthesizeable, MuxD scan design
  - ABIST/MBIST for arrays
  - Built-in LBIST unit

13.4 Microarchitecture summary

The e200z710n3, e200z709, and e200z425Bn3 processor cores utilize a five-stage instruction pipeline with two stages for execution. The Instruction Fetch, Instruction Decode/Register File Read/EA Calc, Execute0/Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single-clock instruction execution for most instructions. Figure 13-1 shows a block diagram of the e200z architecture.

Each integer execution unit consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel Shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register Manipulation Unit (CRU), a Count Leading Zeros unit (CLZ), and result feed-forward hardware. Integer EU1 also supports hardware division and a 32 x 32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single-cycle with the exception of the multiply, which is implemented with a pipelined hardware array, and divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incrementer and dedicated branch address adders to minimize delays during change-of-flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clock cycles for correctly predicted branches. Conditional branches that are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal. These instructions can be pipelined to allow effective single-cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to optimize effective address generation.

The CRU supports the condition register (CR) and condition register operations defined by Power Architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

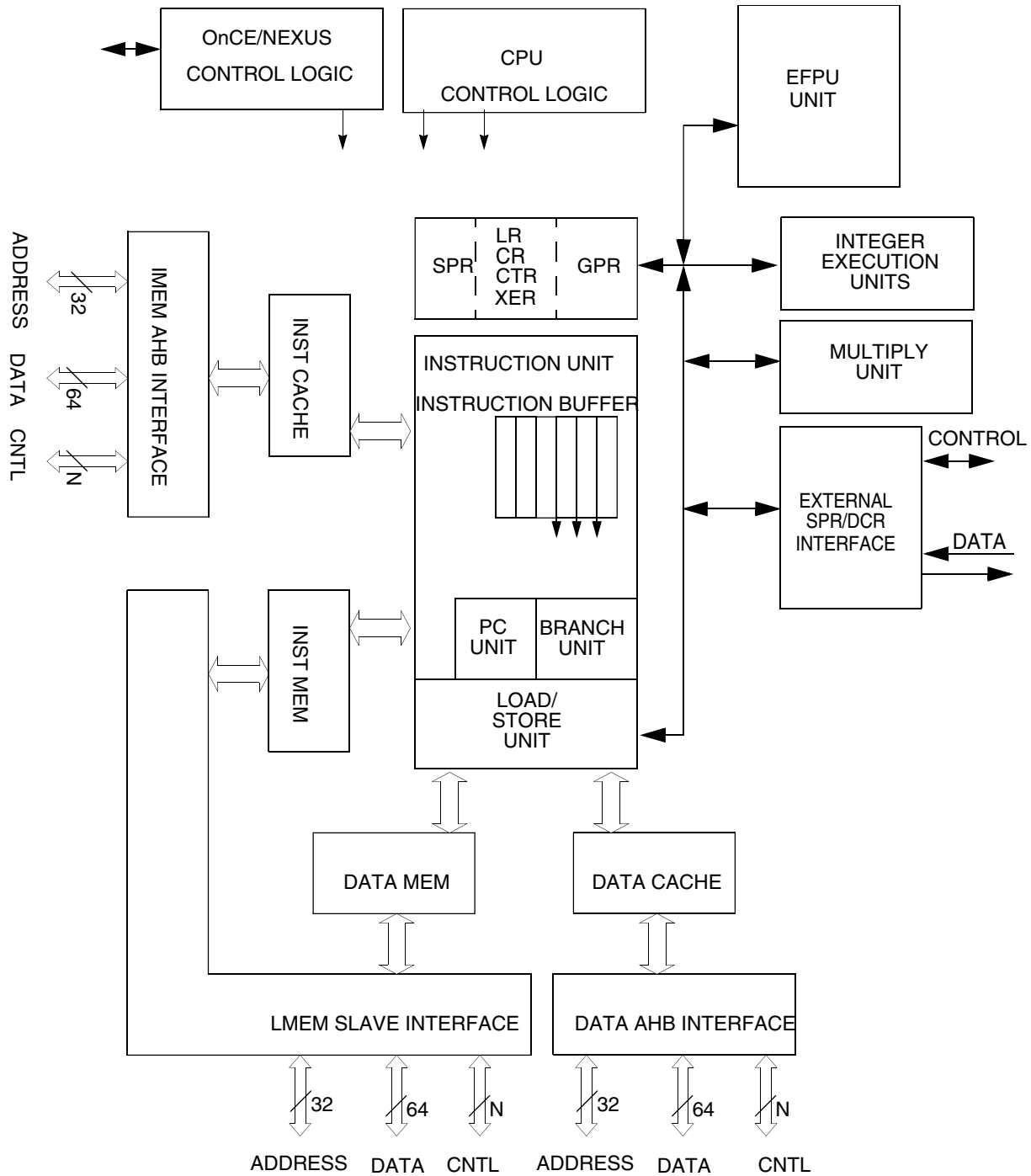


Figure 13-1. e200z block diagram

Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The LSP APU (included only on the e200z425Bn3 core) supports vector instructions operating on 16-bit and 32-bit fixed-point data types. The EFPU2 APU supports 32-bit IEEE-754 single-precision floating-point formats and operates in a pipelined fashion. The

32-bit general-purpose register file is used for source and destination operands. There is a unified storage model for scalar single-precision floating-point data types of 32 bits and the normal integer type. Low latency floating-point add, subtract, mixed add/subtract, sum, diff, min, max, multiply, multiply-add, multiply-sub, divide, square root, compare, and conversion operations are provided. Most operations can be pipelined.

13.4.1 Instruction unit features

The features of the e200z710n3, e200z709, and e200z425Bn3 instruction units are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds as many as eight 32-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) supporting single-cycle execution of successfully predicted branches

13.4.2 Integer unit features

The e200z710n3, e200z709, and e200z425Bn3 integer units support single-cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the Count Leading Zeros function
- 32-bit single-cycle Barrel Shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-14 clock cycles with minimized execution timing (EU1 only)
- Pipelined 32 x 32 hardware multiplier array supports 32 x 32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

13.4.3 Load/Store unit features

The e200z710n3, e200z709, and e200z425Bn3 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations

- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions

13.4.4 EFPU2 Floating-Point Unit Features

The EFPU2 embedded floating-point unit supports pipelined operation of most floating-point operations, allowing a throughput of one FP operation per cycle, and supports a variety of operations:

- Supports IEEE754 single-precision data format
- Supports IEEE754 half-precision format for conversion <-> single-precision format
- EFPU2 instructions utilize the 32-bit GPRs for minimized context save/restore overhead
- Provides low latency pipelined floating-point add, subtract, add/sub, sub/add, multiply, multiply-add, min, max, absolute value, compare, and conversion instructions with single-cycle throughput
- Default results mode for real-time applications allows for exception-free operations when underflow or overflows occur
- Trap Enable controls allow for trapping of various operation exceptions and emulation of IEEE754 boundary case results
- Floating-point Divide and Square root logic operating in 13 (FP div) and 15 (FP sqrt) clock cycles

13.4.5 LSP Lightweight Signal Processing Unit Features

The LSP APU is designed to accelerate signal processing applications normally suited to DSP operation.

- LSP instructions operate on the 32-bit GPRs
- Supports SIMD integer operations - arithmetic, logical, shift
- Supports SIMD rounding, saturation, pack, unpack, merge, extract, select for efficient data manipulation
- Supports SIMD multiply/multiply accumulate/ dot product operations
- 16, 32, and (limited) 64-bit signed and unsigned integer data types
- 16-bit and 32-bit signed fractional support
- Guarded 32-bit and 64-bit signed fractional support
- Single-cycle throughput for all instructions (except fractional divide)
- Supports fractional divide operations (32/32->32)

- Specialized operations such as field manipulation, count-leading,
- Supports specialized load/store operations with integrated data manipulation
- Multiple addressing modes supported including circular and bit-reversed addressing
- Sustained throughput of up to 2 1 16x16-bit multiply/accumulate operations per cycle.

13.4.6 MPU features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support from 1 byte to 4 GB
- 8-bit process identifier
- Bypass capability for individual access types
- Maskable process identifier and address
- Programmable memory attributes
- Entry invalidation protection
- Write-once option
- Alternate Debug Breakpoint/Watchpoint function

13.4.7 Cache features

The features of the caches are as follows:

- 2-Way Set Associative Harvard Instruction and Data Caches
  - e200z710-16 KB ICache, 4 KB DCache
  - e200z425-8 KB ICache, No DCache
  - e200z709-No ICache, No DCache
- Writethrough support
- 8-entry Store Buffer
- Linefill Buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Way locking support
- Write allocation policies

- Support for multi-bit EDC with correction/auto-invalidation capability for the instruction and data caches
- Hardware debug cache invalidate support for the data cach evia DBL1CCSR0 register
- Software and hardware debug access to cache tag, status, and data storage via CDANTL and CDADATA dedicated control registers

13.4.8 Local memory features

The features of the Local Instruction and Data Memories are as follows:

- Local Instruction and Data Memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z710n3-16 KB IMEM, 64 KB DMEM
  - e200z425Bn3-16 KB IMEM, 64 KB DMEM
  - e200z709-No IMEM, No DMEM
- I-MEM provides RAM storage for instructions allowing the CPU to execute these instructions with 0 wait-state delay.
- I-MEM includes a feature to allow the memory to be relocated in the memory map as seen by the local CPU, such that it overlays MCU code flash (The memory map to other bus masters is unchanged). This relocation of I-MEM into the flash region allows IMEM to be accessed directly from any location in flash with a single short branch instruction
- IMEM and DMEM are not duplicated on e200z709 core.
- Supports zero wait-state access relative to cache hit timings
- Supports multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC RMW Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and (for DMEM only) 64-bit write data bus

13.4.9 Performance Monitor Unit Features

The features of the e200z710n3 performance monitor include:

- Four configurable 32-bit performance monitor counters each capable of counting selected CPU subsystem events
- Performance Monitor interrupt
- Ability to configure performance monitor resources for debugger use
- Hardware input signals for qualification of counting by individual counters
- Hardware output signals to indicate counter overflows
- Dedicated watchpoint outputs for each counter with programmable periodicity
- Trigger On/Off control for each counter based on subsystem events
- Multiple selectable event types including # of clock cycles, CPU stall cycles, # of instructions completed, # of interrupts taken, # of memory accesses (by type), # of instruction or data cache misses, # of cycles with 0, 1, or 2 instructions issued, # of instructions in a given class completed (ldst, branch, integer, FP, etc.), cache linefills, and many other event types

13.4.10 e200z710n3 and e200z425Bn3 system bus features

The features of the e200z710n3 and e200z425Bn3 system bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Data interface provides separate uni-directional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

13.4.11 e200z709 system features

Checker Core_0 is implemented using an e200z709 core. When enabled, Checker Core_0s is used in Lock Step with Main_Core_0. The e200z709 core can only be enabled or disabled by the user. For the MPC5777M device, the Checker Core_0s has no system bus and cannot independently execute user code. The core is only used to execute the same instruction stream as Main Core 0.

The e200z709 core is identical to the e200z710n3 with five exceptions:

- IMEM memory array is not present
- ICache memory array is not present
- DMEM memory array is not present
- DCache memory array is not present
- The e200z709 has a pair of system bus interfaces, which are driven/sampled by the delayed Lock Step logic

13.5 Availability of detailed documentation

Detailed documentation of the e200z4xxd and e200z7xxd cores are provided in separate core reference manuals (CRMs). The CRMs are available online at http://www.freescale.com.

Chapter 14
Core description

14.1 Overview of the e200z425Bn3 core

The e200z425Bn3 is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z425Bn3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed-point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs support register pairing in order to support vector instructions defined by the LSP APU. These instructions operate on a vector pair of 16-bit or 32-bit data types, and deliver 16-bit vector and 32-bit scalar results. A wide variety of data manipulation, multiply, multiply/accumulate, and dot product instructions along with specialized memory access instructions allow a sustained throughput of up to two 16x16-bit multiply/accumulate operations per cycle.

The e200z425Bn3 core integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z425Bn3 core contains an 8 KB Instruction Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

14.2 Register model

This section describes the registers implemented in the e200z425Bn3 core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

e200z425Bn3 extends usage of the general purpose registers to support EFPU /LSP APU operations on the 32-bit GPR registers.

Figure 14-1 and Figure 14-2 show the complete e200z425Bn3 register set, all of which are accessible in supervisor mode. Figure 14-3 and Figure 14-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

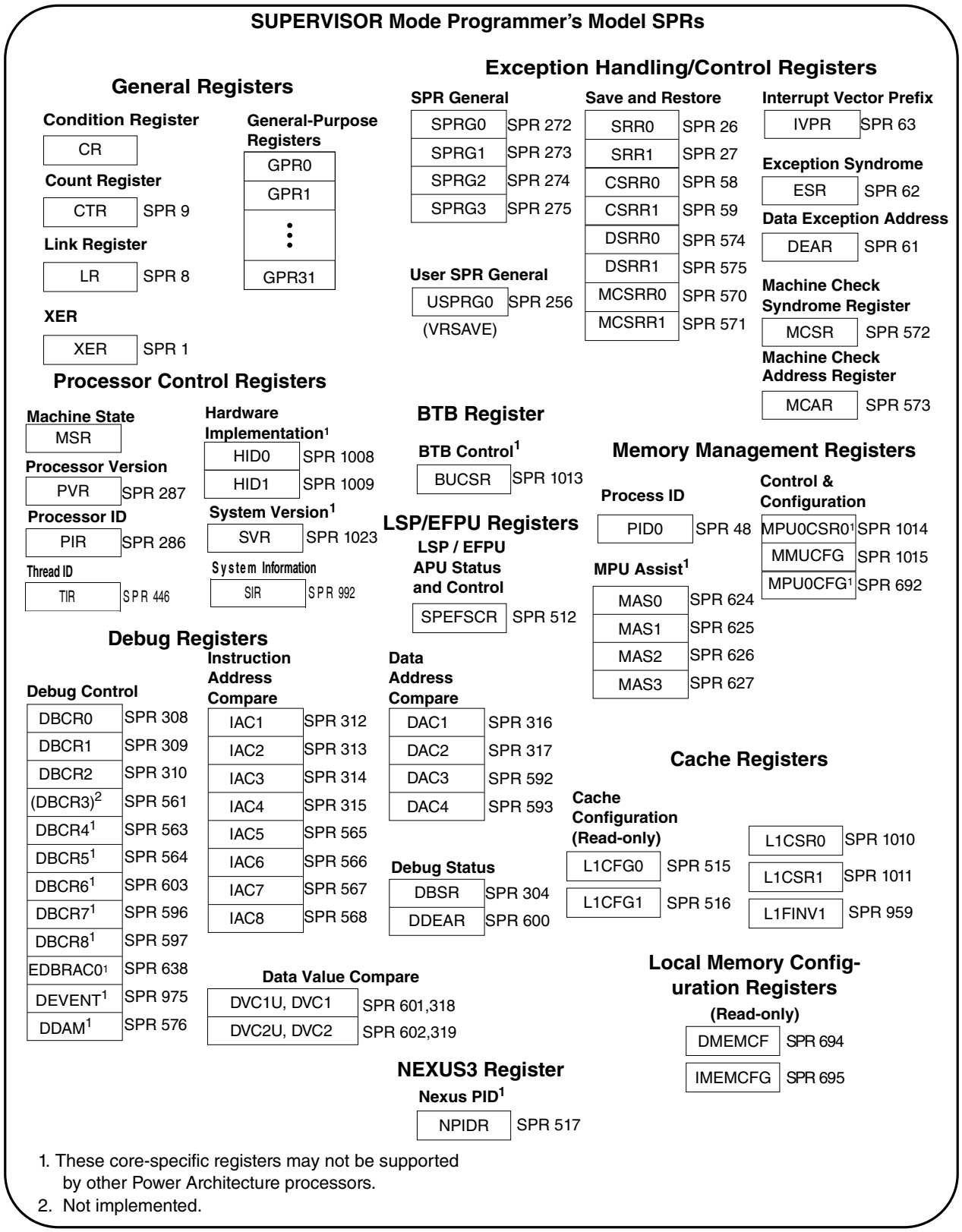


Figure 14-1. e200z425Bn3 Supervisor Mode Programmer's Model SPRs

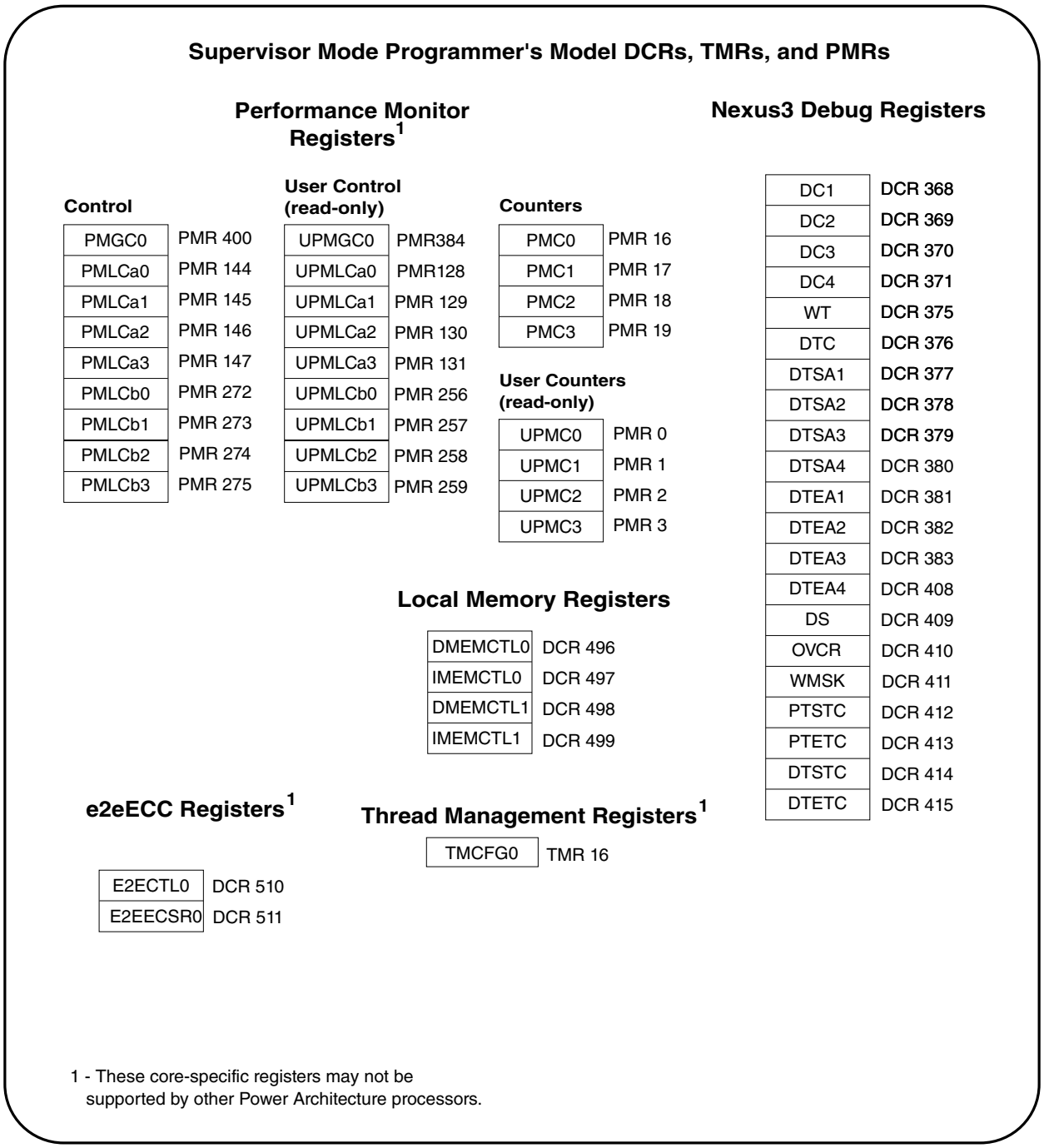


Figure 14-2. e200z425Bn3 Supervisor Mode Programmer's Model DCRs and PMRs

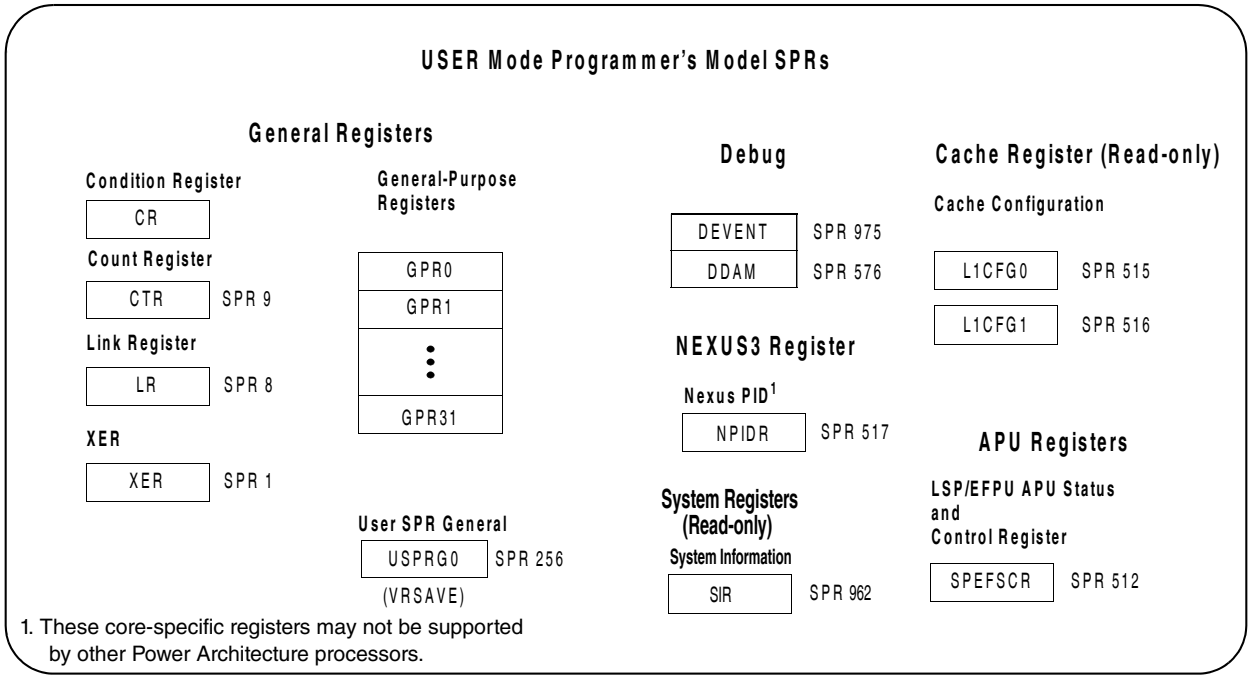


Figure 14-3. e200z425Bn3 User Mode Programmer's Model SPRs

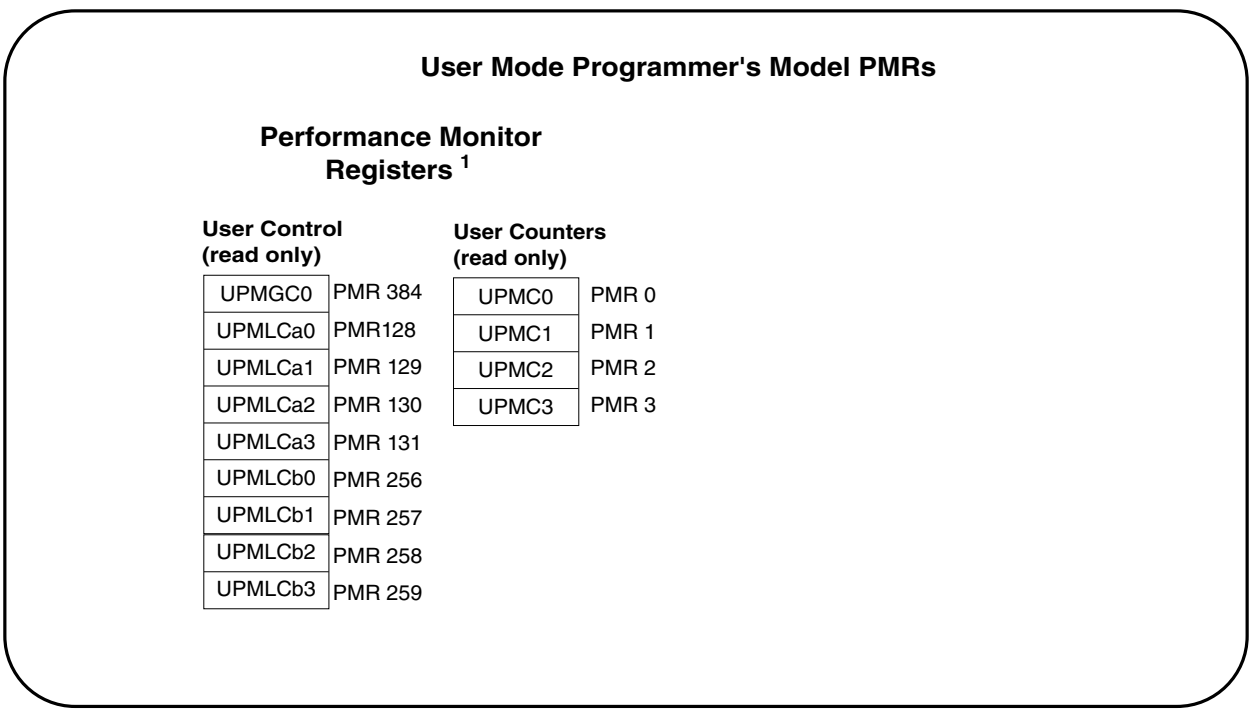


Figure 14-4. e200z425Bn3 User Mode Programmer's Model PMRs

14.2.1 Hardware Implementation Dependent Register 0 (HID0)

The HID0 register is a core implementation dependent register used for various configuration and control functions. It is shown in the following figure.

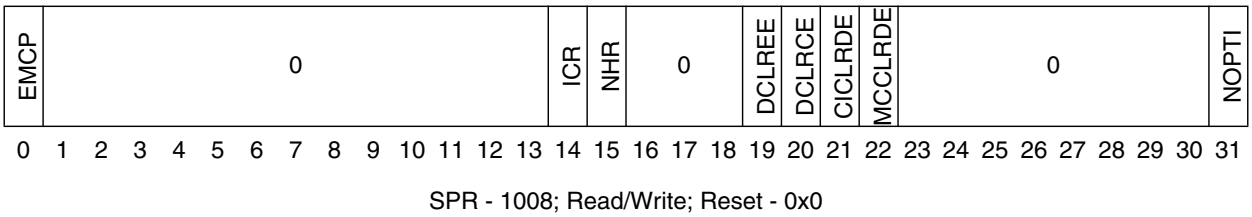


Figure 14-5. Hardware Implementation Dependent Register 0 (HID0)

The HID0 fields are defined in the following table.

Table 14-1. Hardware Implementation-Dependent Register 0

Bits	Name	Description
0 [32]	EMCP	Enable machine check pin (p_mcp_b) 0 p_mcp_b pin is disabled. 1 p_mcp_b pin is enabled. Asserting p_mcp_b causes a machine check interrupt to be reported.
1:13 [33:45]	-	Reserved
14 [46]	ICR	Interrupt Inputs Clear Reservation 0 External Input, Critical Input, and Non-Maskable Interrupts do not affect reservation status 1 External Input, Critical Input, and Non-Maskable Interrupts clear an outstanding reservation
15 [47]	NHR	Not hardware reset 0 indicates to a reset exception handler that a reset occurred if software had previously set this bit 1 indicates to a reset exception handler that no reset occurred if software had previously set this bit Provided for software use. Set anytime by software, cleared by reset.
16:18 [48:50]	-	Reserved
19 [51]	DCLREE	Debug Interrupt Clears MSR_{EE} 0 MSR_{EE} unaffected by Debug Interrupt 1 MSR_{EE} cleared by Debug Interrupt This bit controls whether Debug interrupts force External Input interrupts to be disabled, or whether they remain unaffected.
20 [52]	DCLRCE	Debug Interrupt Clears MSR_{CE} 0 MSR_{CE} unaffected by Debug Interrupt 1 MSR_{CE} cleared by Debug Interrupt This bit controls whether Debug interrupts force Critical interrupts to be disabled, or whether they remain unaffected.
21 [53]	CICLRDE	Critical Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Critical class interrupt

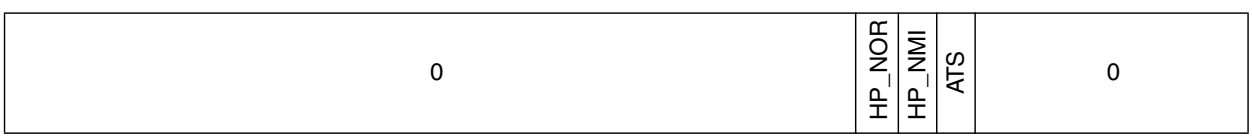
Table continues on the next page...

Table 14-1. Hardware Implementation-Dependent Register 0 (continued)

Bits	Name	Description
		1 MSR_{DE} cleared by Critical class interrupt This bit controls whether Critical interrupts force Debug interrupts to be disabled, or whether they remain unaffected. Machine Check interrupts have a separate control bit. Note that if Critical Interrupt Debug events are enabled ($DBCR0_{CIRPT}$ set), and MSR_{DE} is set at the time of a Critical Input Critical interrupt, a debug event will be generated after the Critical Interrupt Handler has been fetched, and the Debug handler will be executed first. In this case, $DSRR0_{DE}$ will have been cleared, such that after returning from the debug handler, the Critical interrupt handler will not be run with MSR_{DE} enabled.
22 [54]	MCCLRDE	Machine Check Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Machine Check interrupt 1 MSR_{DE} cleared by Machine Check interrupt This bit controls whether Machine Check interrupts force Debug interrupts to be disabled, or whether they remain unaffected.
23:30 [55:62]	-	Reserved
31 [63]	NOPTI	No-op Touch Instructions 0 icbt, dcbt, dcbtst instructions operate normally 1 icbt, dcbt, dcbtst instructions are no-oped This bit only affects the icbt, dcbt, and dcbtst instructions.

14.2.2 Hardware Implementation Dependent Register 1 (HID1)

The HID1 register is used for bus configuration and system control. It is shown in the following figure.



SPR - 1009; Read/Write; Reset - 0x0

Figure 14-6. Hardware Implementation Dependent Register 1 (HID1)

The HID1 fields are defined in the following table.

Table 14-2. Hardware Implementation-Dependent Register 1

Bits	Name	Description
0:21 [32:53]	-	Reserved

Table continues on the next page...

Table 14-2. Hardware Implementation-Dependent Register 1 (continued)

Bits	Name	Description
22 [54]	HP_NOR	High priority elevation for normal and external interrupts 0 - High priority elevation for normal and external interrupts is disabled 1 - High priority elevation for normal and external interrupts is enabled The HP_NOR bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided. These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
23 [55]	HP_NMI	High priority elevation for NMI and critical interrupts 0 - High priority elevation for NMI and critical interrupts is disabled 1 - High priority elevation for NMI and critical interrupts is enabled The HP_NMI bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
24 [56]	ATS	Atomic status (read-only) Indicates state of the reservation bit in the load/store unit.
25:31 [57:63]	-	Reserved

14.3 Dual-issue operation

The instruction issue unit attempts to issue two instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. As many as three results can be simultaneously written.

14.4 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in Table 14-3. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during execution.

Timing for EFPU instructions is detailed in Table 14-6.

Timing for LSP instructions is detailed in:

- LSP simple instruction timing—Table 14-7
- LSP complex instruction timing—Table 14-8
- LSP shift/rotate instruction timing—Table 14-9
- LSP vector compare instruction timing—Table 14-10
- LSP vector select instruction timing—Table 14-11
- LSP vector data arrangement instruction timing—Table 14-12
- LSP multiply and multiply/accumulate instruction timing—Table 14-13
- LSP dot product instruction timing—Table 14-14
- LSP miscellaneous vector instruction timing—Table 14-15
- LSP load and store instruction timing—Table 14-16

Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 14-3. Instruction class cycle counts

Class of Instructions	Latency	Throughput	Special notes
integer: **add**, **sub**, **shift**, **rotate**, **logical**, **cntlzw**	1	1	—
integer: **compare**	1	1	—
Branch	3/2/1	3/2/1	Correct branch lookahead allows single-cycle execution. Worst-case mispredicted branch is 3 cycles.
multiply (saturating/non-saturating)	2	1	—
divide	4–14	4–14	Data-dependent timing
CR logical	1	1	—
loads (non-multiple)	2	1	—
load multiple	2 + n/2 (max)	2 + n/2 (max)	Actual timing depends on *n* and address alignment.
stores (non-multiple)	2	1	—

Table continues on the next page...

Table 14-3. Instruction class cycle counts (continued)

Class of Instructions	Latency	Throughput	Special notes
store multiple	2 + n/2 (max)	2 + n/2 (max)	Actual timing depends on *n* and address alignment.
mtmsr, **wrtee**, **wrteei**	3&	3&	—
mcrf	1	1	—
mfspr, **mtspr**	4&	4&	Applies to Debug SPRs, optional unit SPRs
mfspr, **mfmsr**	1	1	Applies to certain limited internal, non Debug SPRs
mfcr, **mtcr**	1	1	—
se_rfi, **se_rfci**, **se_rfdi**, **se_rfmci**	3	—	—
se_sc, **e_sc**	4	—	—
e_tw	4	—	Trap taken timing

Detailed timing for each instruction mnemonic along with serialization requirements is shown in Table 14-4 and Table 14-5.

Table 14-4. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	3/2/1	—
se_bclri	1	—
se_bctr	3/2	—
se_bctrl	3/2	—
se_bgeni	1	—
se_bl	3/2/1	—
se_blr	3/2	—
se_blrl	3/2	—
se_bmaski	1	—
se_b	3/2/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—

Table continues on the next page...

Table 14-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_cmpl	1	—
se_cmpli	1	—
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	2	—
se_lhz	2[2]	—
se_li	1	—
se_lwz	2[2]	—
se_mfar	1	—
se_mfctr	1	—
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	3	Refetch
se_rfdi	3	Refetch
se_rfi	3	Refetch
se_rfmci	3	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	2	—
se_sth	2[2]	—
se_stw	2[2]	—

Table continues on the next page...

Table 14-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_sub	1	—
se_subf	1	—
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 14-5. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]	1	—
adde[.]	1	—
addeo[.]	1	—
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	3/2/1	—
e_bc	3/2/1	—
e_bcl	3/2/1	—
e_bl	3/2/1	—
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—

Table continues on the next page...

Table 14-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
cmpl	1	—
e_cmpl16i	1	—
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—
dcbst	1	—
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	4–14[1]	—
divwo[.]	4–14[1]	—
divwu[.]	4–14[1]	—
divwuo[.]	4–14[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	2	—
e_lbz	2	—
e_lbzu	2	—
lbzux	2	—
lbzx	2	—
e_lha	2[2]	—
lharx	2	—
e_lhau	2[2]	—

Table continues on the next page...

Table 14-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
lhaux	2[2]	—
lhax	2[2]	—
lhbrx	2[2]	—
e_lhz	2[2]	—
e_lhzu	2[2]	—
lhzux	2[2]	—
lhzx	2[2]	—
e_li	1	—
e_lis	1	—
e_lmw	2+(n/2)	—
lwarx	2	—
lwbrx	2[2]	—
e_lwz	2[2]	—
e_lwzu	2[2]	—
lwzux	2[2]	—
lwzx	2[2]	—
mbar	1[3]	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	4[3]	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	3[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	1	None
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2	—
mulhwu[.]	2	—
e_mull2i	2	—
e_mulli	2	—
mullw[.]	2	—

Table continues on the next page...

Table 14-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
mullwo[.]	2	—
nand[.]	1	—
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	4	Refetch
slw[.]	1	—
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	2	—
stbcx.	2	—
e_stbu	2	—
stbux	2	—
stbx	2	—
e_sth	2[2]	—
sthbrx	2[2]	—
sthcx.	2	—
e_sthu	2[2]	—
sthux	2[2]	—
sthx	2[2]	—
e_stmw	2 + (*n*/2)	—
e_stw	2[2]	—
stwbrx	2[2]	—
stwcx.	2	—
e_stwu	2[2]	—
stwux	2[2]	—
stwx	2[2]	—

Table continues on the next page...

Table 14-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
subf[.]	1	—
subfc[.]	1	—
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	3	Completion
wrteei	3	Completion
xor[.]	1	—
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

Instruction timing for EFPU single-precision scalar floating-point instructions is shown in Table 14-6. The table is sorted by opcode.

Table 14-6. EFPU single-precision scalar floating-point instruction timing

Instruction	Latency	Throughput	Comments
efsabs	2	1	—
efsadd	2	1	—
efscfh	2	1	—
efscfsf	2	1	—
efscfsi	2	1	—
efscfuf	2	1	—
efscfui	2	1	—
efscmpeq	2	1	—
efscmpgt	2	1	—
efscmplt	2	1	—
efscth	2	1	—
efsctsf	2	1	—
efsctsi	2	1	—
efsctsiz	2	1	—

Table continues on the next page...

Table 14-6. EFPU single-precision scalar floating-point instruction timing (continued)

Instruction	Latency	Throughput	Comments
efsctuf	2	1	—
efsctui	2	1	—
efsctuiz	2	1	—
efsdiv	13	13	Blocking, no execution overlap with next instruction
efsmadd	2	1[1]	Destination also used as source
efsmsub	2	1[1]	Destination also used as source
efsmax	2	1	—
efsmin	2	1	—
efsmul	2	1	—
efsnabs	2	1	—
efsneg	2	1	—
efsnmadd	2	1[1]	Destination also used as source
efsnmsub	2	1[1]	Destination also used as source
efssqrt	11	11	Blocking, no overlap with next instruction
efssub	2	1	—
efststeq	2	1	—
efststgt	2	1	—
efststlt	2	1	—

1. Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.

Instruction timing in number of processor clock cycles for LSP instructions are shown in the following tables. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during divide execution.

Instruction timing for LSP simple instructions is shown in Table 14-7.

Table 14-7. LSP simple instruction timing

Basic operation	Variants	Latency	Throughput
Absolute value	**zvabsh**, **zabsw**	1	1
	zvabshs, **zabsws**	1	1
Add	**zaddd**, **zvaddh**, **zvaddw**	1	1
	zadddss, **zvaddhss**, **zaddwss**, **zvaddwss**, **zadddus**, **zvaddhus**, **zaddwus**	1	1
	zvaddhx, **zvaddhxss**	1	1
	zaddhe[s,u]w, **zaddho[s,u]w**	1	1
	zvaddih	1	1
	zaddwgsf	1	1

Table continues on the next page...

Table 14-7. LSP simple instruction timing (continued)

Basic operation	Variants	Latency	Throughput
	zaddwgsi, **zaddwgui**	1	1
AddSubf	**zvaddsubfh**, **zvaddsubfhss**, **zvaddsubfw**, **zvaddsubfwss**	1	1
	zvaddsubfhx, **zvaddsubfhxss**	1	1
Count Leading	**zvcntlsh**, **zvcntlzh**, **zcntlsw**	1	1
Negate	**zvnegh**	1	1
	zvneghs, **znegws**	1	1
	zvnegho, **zvneghos**	1	1
Round	**zrndwh**, **zrndwhss**	1	1
Saturate	**zsatsdsw**, **zsatsduw**	1	1
	zsatuduw	1	1
	zvsatshuh	1	1
	zvsatuhsh	1	1
	zsatswsh, **zsatswuh**	1	1
	zsatswuw	1	1
	zsatuwsh, **zsatuwuh**	1	1
	zsatuwsw	1	1
Subf	**zsubfd**, **zvsubfh**, **zvsubfw**	1	1
	zsubfdss, **zvsubfhss**, **zsubfwss**, **zvsubfwss**, **zsubfdus**, **zvsubfhus**, **zsubfwus**, **zvsubfwus**	1	1
	zvsubfhx, **zvsubfhxss**	1	1
	zsubfhe[s,u]w, **zsubfho[s,u]w**	1	1
	zvsubifh	1	1
	zsubfwgsf	1	1
	zsubfwgsi, **zsubfwgui**	1	1
SubfAdd	**zvsubfaddh**, **zvsubfaddhss**, **zvsubfaddw**, **zvsubfaddwss**	1	1
	zvsubfaddhx, **zvsubfaddhxss**	1	1

Instruction timing for LSP complex instructions is shown in Table 14-8. For the divide instruction, the number of stall cycles is (latency) for following instructions.

Table 14-8. LSP complex instruction timing

Operation	Instruction	Latency	Throughput
Divide	**zdivwsf**	4 – 14[1]	4 – 14[1]

1. Timing is data-dependent.

Instruction timing for LSP shift/rotate instructions is shown in Table 14-9.

Table 14-9. LSP shift/rotate instruction timing

Basic operation	Variants	Latency	Throughput
Logical Shift Left	**zvslh**, **zvslhi**	1	1
Logical Shift Right	**zvsrhu**, **zvsrhiu**	1	1
Rotate Left	**zvrlh**, **zvrlhi**	1	1
Signed Shift Left and Saturate	**zslwss**, **zslwiss**, **zvslhss**, **zvslhiss**	1	1
Unsigned Shift Left and Saturate	**zslwus**, **zslwius**, **zvslhus**, **zvslhius**	1	1
Arithmetic Shift Right	**zvsrhs**, **zvsrhis**	1	1

Instruction timing for LSP vector compare instructions is shown in Table 14-10.

Table 14-10. LSP vector compare instruction timing

Basic comparison operation	Instruction	Latency	Throughput
=	**zvcmpeqh**>	1	1
>	**zvcmpgths**, **zvcmpgthu**	1	1
<	**zvcmplthu**, **zvcmplthu**	1	1

Instruction timing for LSP vector select instructions is shown in Table 14-11.

Table 14-11. LSP vector select instruction timing

Operation	Instruction	Latency	Throughput
Select	**zvselh**	1	1

Instruction timing for LSP vector data arrangement instructions is shown in Table 14-12.

Table 14-12. LSP vector data arrangement instruction timing

Operation	Instruction	Latency	Throughput
Merge	**zvmergehih**	1	1
	zvmergehiloh	1	1
	zvmergeloh	1	1
	zvmergelohih	1	1
Pack	**zvpkswshfrs**	1	1
	zvpkswshs, **zvpkswuhs**, **zvpkuwuhs**	1	1
	zpkswgswfrs	1	1
	zvpkshgwshfrs	1	1
Splat	**zvsplatfih**	1	1
	zvsplatih	1	1
Unpack	**zunpkwgsf**	1	1
	zvunpkhgwsf	1	1
	zvunpkhsf, **zvunpkhsi**, **zvunpkhui**	1	1

Instruction timing for LSP multiply and multiply/accumulate instructions is shown in Table 14-13.

Table 14-13. LSP multiply and multiply/accumulate instruction timing

Instruction	Latency	Throughput
all **z[v]m{h,w}** instructions	—	1

Instruction timing for LSP dot product instructions is shown in Table 14-14.

Table 14-14. LSP dot product instruction timing

Instruction	Latency	Throughput
all **z[v]dotp** instructions	—	1

Instruction timing for LSP miscellaneous instructions is shown in Table 14-15.

Table 14-15. LSP miscellaneous vector instruction timing

Operation	Instruction	Latency	Throughput
Circular increment	**zcircinc**	1	1
Bit reversed increment	**zbrminc**	1	1

Instruction timing for LSP load and store instructions instructions is shown in Table 14-16.

Table 14-16. LSP load and store instruction timing

Instruction	Latency	Throughput
all **zv** loads	—	1
all **zv** stores	—	1

14.5 Reservation instructions and cache interactions

If the CPU supports reservation instruction functionality, the CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

14.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. This register is also used by the LSP APU. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in the following figure.

SPR - 512; Read/Write; Reset - 0x0

Figure 14-7. LSP/EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 14-17. LSP/EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z425Bn3 Supports only mode 0.

Table continues on the next page...

Table 14-17. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
		Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by e200 core)
16 (48)	SOV	Summary integer overflow Defined by LSP APU.
17 (49)	OV	Integer overflow Defined by LSP APU.
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled

Table continues on the next page...

Table 14-17. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
		1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

14.7 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

14.7.1 Cache overview

The e200z425Bn3 processor supports an 8 KB 2-way set-associative instruction cache with a 32-byte line size. The cache improves system performance by providing low-latency data to the e200z425Bn3instruction pipeline, which decouples processor performance from system memory performance.

The e200z425Bn3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

14.7.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in both caches. L1CSR0 implements the WID control field for instruction cache way disabling. The other fields are reserved. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR0 register is shown in the following figure.

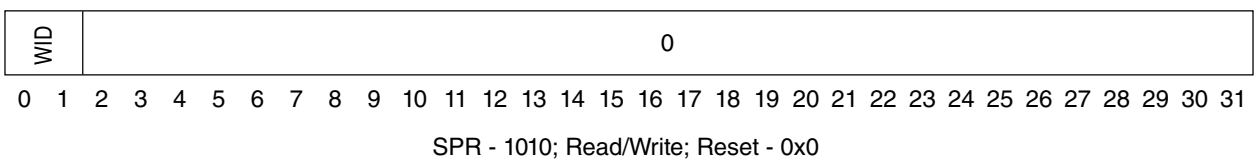


Figure 14-8. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 14-18. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:31	—	Reserved[1]

1. These bits are not implemented and should be written with zero for future compatibility.

14.7.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR1 register is shown in the following figure.

SPR - 1011; Read/Write; Reset - 0x0

Figure 14-9. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 14-19. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits.
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one.

Table continues on the next page...

Table 14-19. L1CSR1 field descriptions (continued)

Bits	Name	Description
		11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses.

Table 14-19. L1CSR1 field descriptions

Bits	Name	Description
		Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

14.7.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z425Bn3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The L1CFG0 register is shown in the following figure.

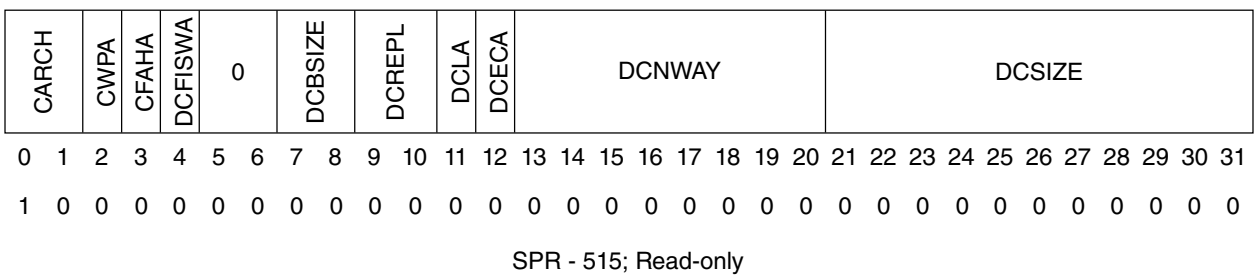


Figure 14-10. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 register indicates a cache architecture value (CARCH) of 2'b10, indicating that an I-Cache but no D-Cache is present.

14.7.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z425Bn3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The L1CFG1 register is shown in the following figure.

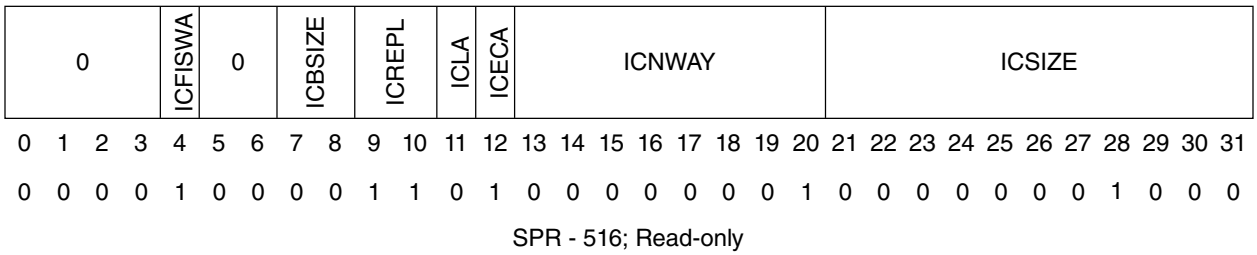


Figure 14-11. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 14-20. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

14.7.6 Cache Invalidate by Set and Way

e200z425Bn3 supports cache set/way invalidation under software control. The instruction cache may be invalidated by index and way through a **mtspr l1finv1** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV1) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when the instruction cache is disabled.

14.7.7 L1FINV1

The L1FINV1 register is shown in following figure.

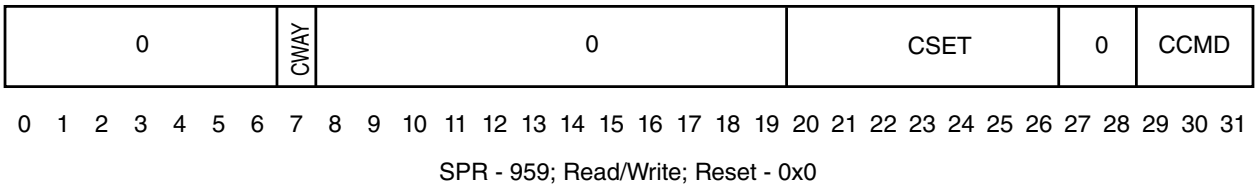


Figure 14-12. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 14-21. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8: 19	—	Reserved[1] for set extension
20 :26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

14.7.8 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of the instruction cache. Seven parity check bits are provided for each tag entry for the tag array to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for

each doubleword in the data arrays of the I-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR1[ICECE] control field. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the instruction cache when cache error checking is disabled ($L1CSR1_{ICECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

14.7.8.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the cache, each cache line in the I-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR1_{ICLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on instruction fetch, accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR1_{ICLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch , if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, cache contents are ignored for lines with those errors. Cache invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable access will perform the lockout function if required.

14.7.9 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction cache.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi** instruction or an L1FINV1 invalidation operation.

14.8 Exceptions

Interrupts implemented in e200z425Bn3 and the exception conditions that cause them are listed in the following table.

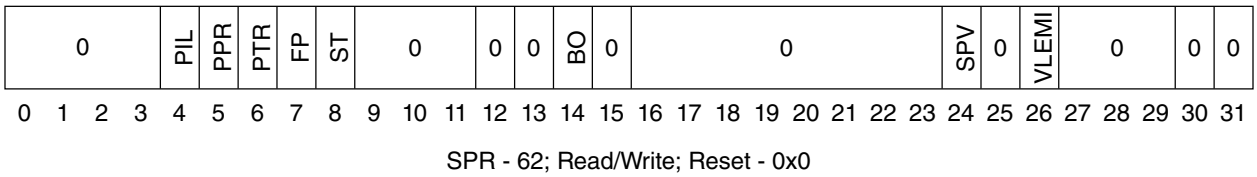
Table 14-22. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. LSP ld and st instructions not properly aligned 4. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1
EFPU Data Exception	0xA0	Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

14.8.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.



SPR - 62; Read/Write; Reset - 0x0

Figure 14-13. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 14-23. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For e200z425Bn3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—

Table continues on the next page...

Table 14-23. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
31	—	Reserved	—

14.8.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z425Bn3 MSR is shown in the following figure.

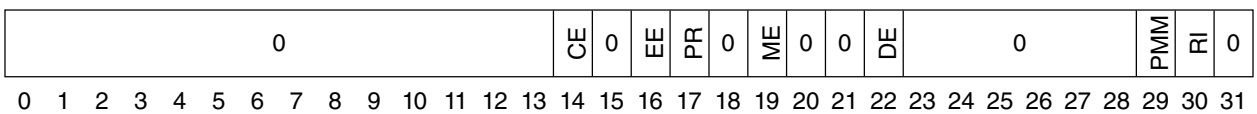


Figure 14-14. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 14-24. MSR field descriptions

Bit(s)	Name	Description
0:13	—	Reserved
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled.

Table continues on the next page...

Table 14-24. MSR field descriptions (continued)

Bit(s)	Name	Description
		1 Debug interrupts are enabled.
23:28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

14.8.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in the following figure.

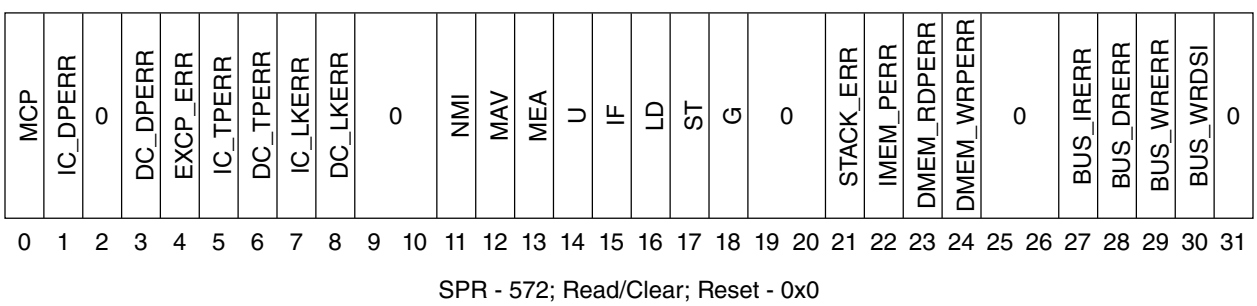


Figure 14-15. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $MSR_{ME} = 1$.

Table 14-25. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR[2]	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR[2]	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1,MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[3]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise

Table continues on the next page...

Table 14-25. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

- "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
- "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

2. Will not be set by e200z425Bn3-1 since no data cache is present.
3. Note that since no address translation mechanism is present on e200z420n3, the MEA value is always set to \0'.

14.8.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

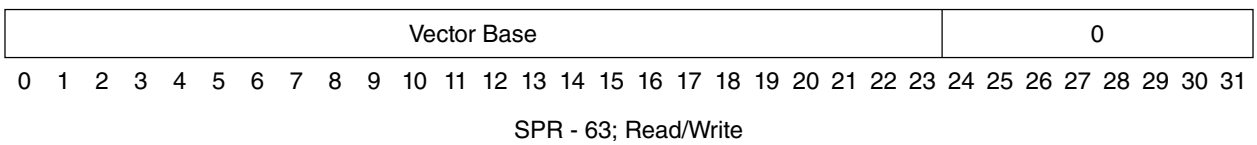


Figure 14-16. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 14-26. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

14.8.5 Interrupt Definitions

14.8.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 14-27. Critical Input Interrupt—register settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0[1]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored)					
	$IVPR_{0:15}$ || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

14.8.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 14-28. Machine Check Interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

14.8.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 14-29. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

14.8.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 14-30. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

14.8.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 14-31. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 14-31. External Input Interrupt—Register Settings (continued)

Register	Setting description					
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

14.8.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.
- Execution of an LSP APU load or store instruction that is not properly aligned.

The following table lists register settings when an alignment interrupt is taken.

Table 14-32. Alignment Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], [SPV], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

14.8.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z425Bn3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z425Bn3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z425Bn3 will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 14-33. Program Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the excepting instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table continues on the next page...

Table 14-33. Program Interrupt—register settings (continued)

Register	Setting description					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			
	Trap:		PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x60					

14.8.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 14-34. Performance Monitor Interrupt—Register Settings

Register	Setting description
SRR0/ DSRR0	Set to the effective address of the next instruction to be executed.
SRR1/	Set to the contents of the MSR at the time of the interrupt.

Table continues on the next page...

Table 14-34. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting description					
DSRR1[1]						
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x70					

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

14.8.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 14-35. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x80					

14.8.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM} = 1$ (internal debug mode) and $MSR_{DE} = 1$).

The following table lists register settings when a Debug interrupt is taken.

Table 14-36. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			
	External Debug Event:		{DEVT1, DEVT2}			
	Performance Monitor Debug Event:		PMI			
	MPU Debug Event:		MPU			
	and optionally, an Imprecise Debug Event flag		{IDE}			
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x90					

1. Assumes that the Debug interrupt is precise

2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

14.8.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 14-37. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

14.8.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 14-38. Embedded Floating-point Round Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 14-38. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description					
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xB0					

14.8.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 14-39. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

14.9 Memory Protection Unit (MPU)

14.9.1 MPU Overview

The e200z425Bn3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

14.9.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

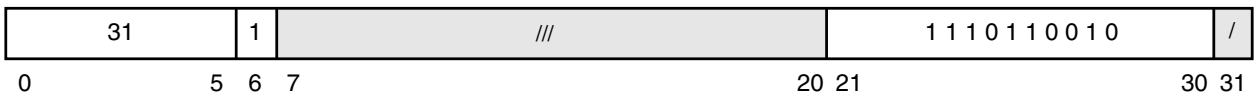
14.9.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

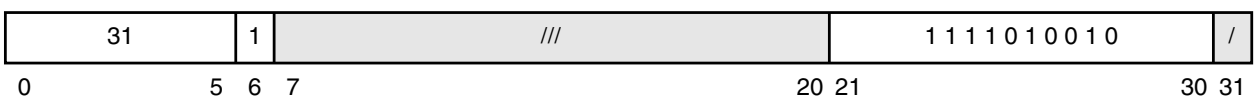
14.9.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

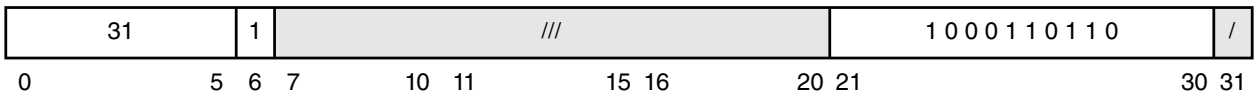
14.9.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the core.

mpusync

MPU Synchronize
mpusync

mpusync



14.9.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z425Bn3 MPU design. The MMUCFG register is shown in the following figure.

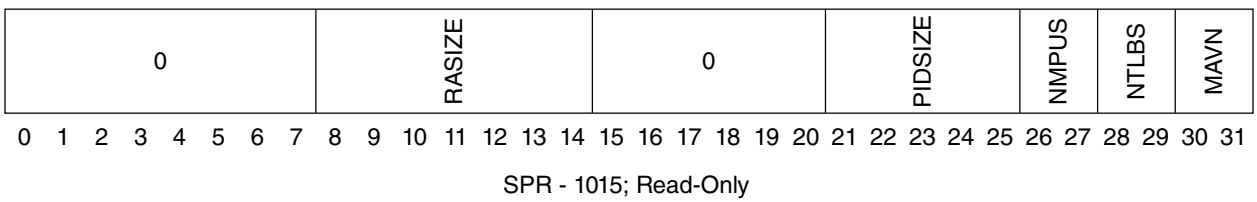


Figure 14-17. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in the following table.

Table 14-40. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

14.9.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z425Bn3 MPU. The MPU0CFG register is shown in following figure.

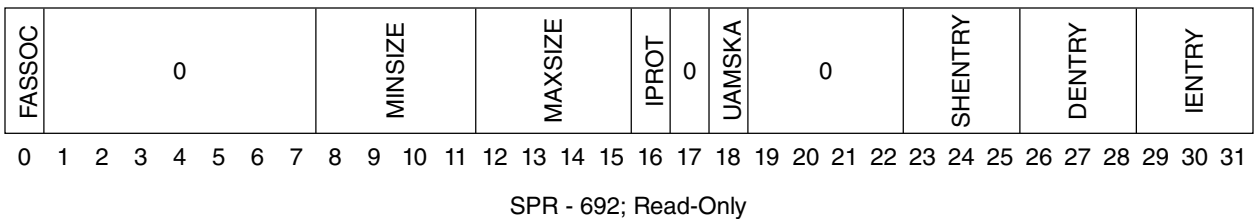


Figure 14-18. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in the following table.

Table 14-41. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

14.9.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in the following figure.

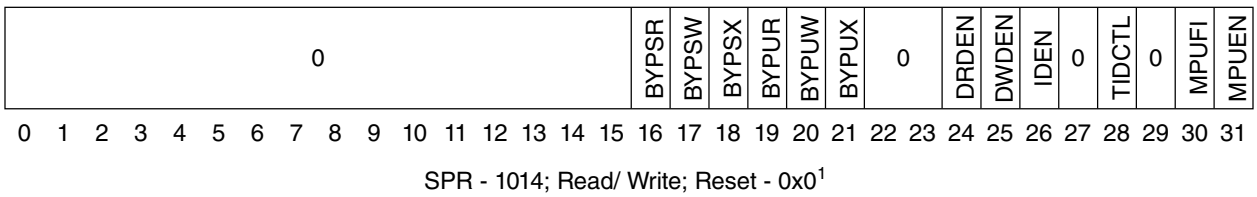


Figure 14-19. MPU0 Control and Status Register 0 (MPU0CSR0)

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM} = 0$, as well as unconditionally by an internal power-on reset signal. If $EDBCR0_{EDM} = 1$, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in the following table.

Table 14-42. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses

Table continues on the next page...

Table 14-42. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYPUW	Bypass MPU protections for User Write accessesThis bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses

Table continues on the next page...

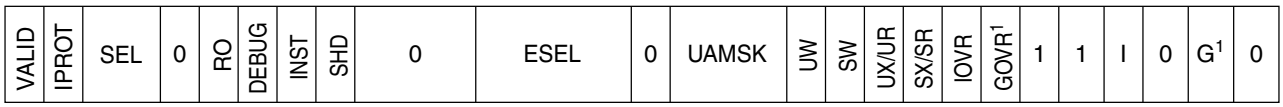
Table 14-42. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation **Note:** Entries that have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

14.9.9 MPU Assist Registers (MAS)

The core uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in Figure 14-20. Fields are defined in Table 14-43.



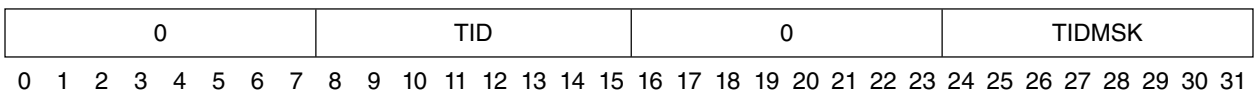
SPR - 624; Read/ Write; Reset - Unaffected

Figure 14-20. MPU Assist Register 0 (MAS0)

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

The MAS1 register is shown in Figure 14-21. Fields are defined in Table 14-43.



SPR - 625; Read/Write; Reset - Unaffected

Figure 14-21. MPU Assist Register 1 (MAS1)

Table 14-43. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx = TID[0] comparison to PID0[0] not masked 1xxxxxxx = TID[0] comparison to PID0[0] is masked x0xxxxxx = TID[1] comparison to PID0[1] not masked x1xxxxxx = TID[1] comparison to PID0[1] is masked xx0xxxxx = TID[2] comparison to PID0[2] not masked xx1xxxxx = TID[2] comparison to PID0[2] is masked xxx0xxxx = TID[3] comparison to PID0[3] not masked xxx1xxxx = TID[3] comparison to PID0[3] is masked xxxx0xxx = TID[4] comparison to PID0[4] not masked xxxx1xxx = TID[4] comparison to PID0[4] is masked xxxxx0xx = TID[5] comparison to PID0[5] not masked xxxxx1xx = TID[5] comparison to PID0[5] is masked xxxxxx0x = TID[6] comparison to PID0[6] not masked xxxxxx1x = TID[6] comparison to PID0[6] is masked xxxxxxx0 = TID[7] comparison to PID0[7] not masked

Table 14-43. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
		xxxxxxx1 = TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in Figure 14-22. Fields are defined in Table 14-44.

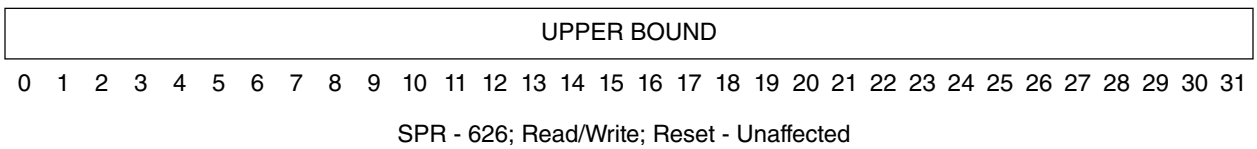


Figure 14-22. MPU Assist Register 2 (MAS2)

Table 14-44. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in Figure 14-23. Fields are defined in Table 14-45.

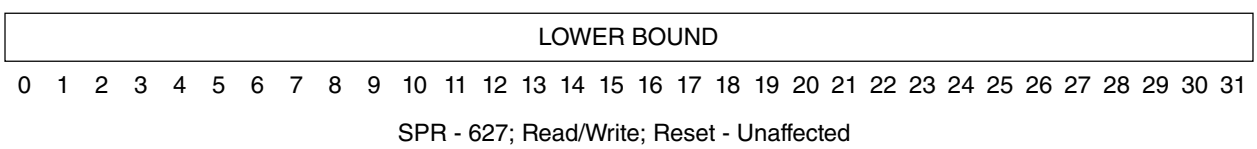


Figure 14-23. MPU Assist Register 3 (MAS3)

Table 14-45. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

14.9.10 MAS Registers Summary

The MAS registers are summarized in the following figure.

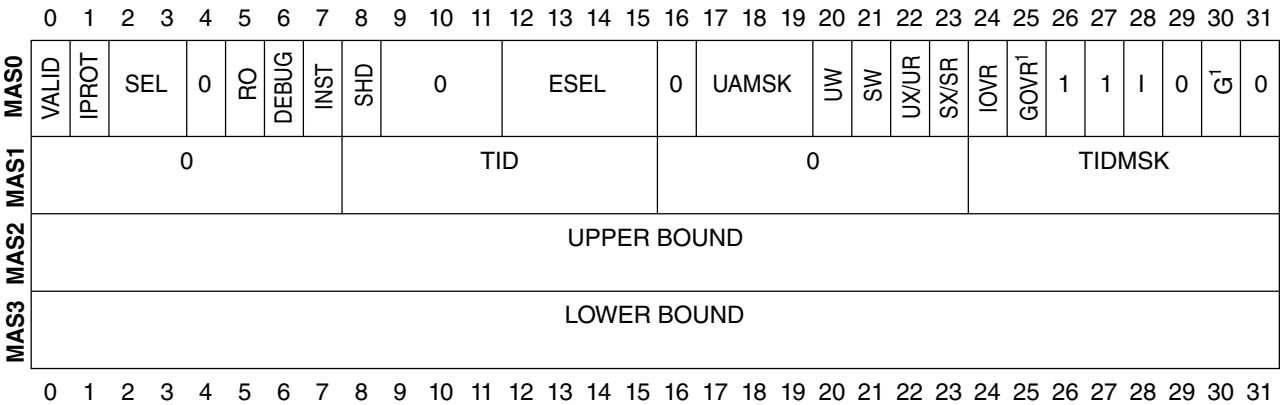


Figure 14-24. MPU Assist Registers Summary

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

14.10 Local memories

14.10.1 Local Instruction and Data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added. This provides low latency access (zero wait-states) similar to an instruction or data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity “tightly coupled” storage capability.

14.10.2 Local memory control and configuration

DMEM and IMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mfspr** instructions. DMEM and IMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

14.10.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z425Bn3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in the following figure.

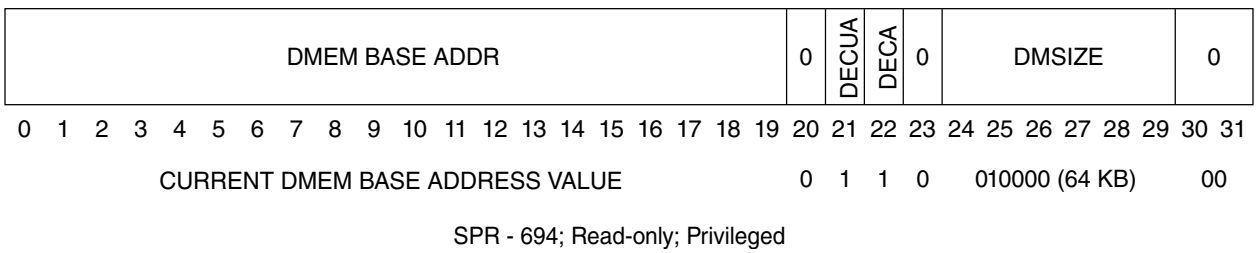


Figure 14-25. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in the following table.

Table 14-46. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0s when the DMEM size is > 4 KB
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB.
30:31	—	Reserved

14.10.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

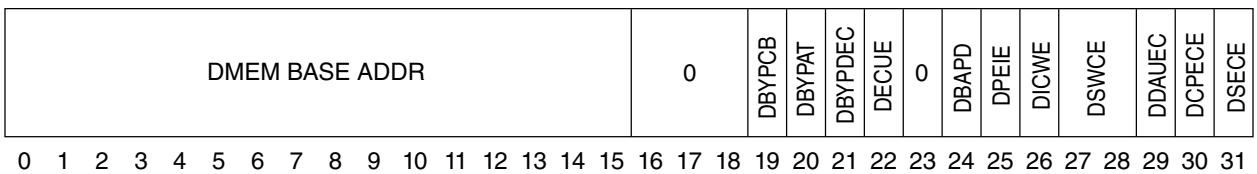


Figure 14-26. DMEM Control Register 0 (DMEMCTL0)

Note

[1] Reset by an internal power-on reset signal or by an internal destructive reset signal. Unaffected by **p_reset_b**.

Table 14-47. DMEMCTL0 field descriptions

Bits	Name	Description
0:15	DMEM BASEADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
16 :18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[8]** input, which is internally hard coded.
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by

Table continues on the next page...

Table 14-47. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[6]** input, which is internally hard coded.
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[7]** input, which is internally hard coded.
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[0]** input which is internally hard coded.
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled.

Table continues on the next page...

Table 14-47. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits.
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1-, 2-, or 3-byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the values of the **p_dmem_rstcfg[1:2]** inputs, which are internally hard coded.
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[3]** input, which is an internally hardcoded signal.
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[4]** input, which is internally hard coded.
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written.

Table 14-47. DMEMCTL0 field descriptions

Bits	Name	Description
		1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[5]** input, which is an internally hard coded signal.

14.10.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed. All other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB,\ DBYPATF,\ DBYPDEC}$.

DSWAC3	DSWAC2	DSWAC1	DSWAC0	DSRAC3	DSRAC2	DSRAC1	DSRAC0	DUWAC3	DUWAC2	DUWAC1	DUWAC0	DURAC3	DURAC2	DURAC1	DURAC0
0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31

DCR - 498; Read/Write; Reset[1] - 32'b0; Privileged

Figure 14-27. DMEM Control Register 1 (DMEMCTL1)

Note

[1] Reset from an internal power-on reset signal or by an internal reset signal.

Table 14-48. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 14-48. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11

Table continues on the next page...

Table 14-48. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM

Table continues on the next page...

Table 14-48. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 = User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 = User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 = User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 = Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 14-48. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

14.10.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z425Bn3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in the following figure.

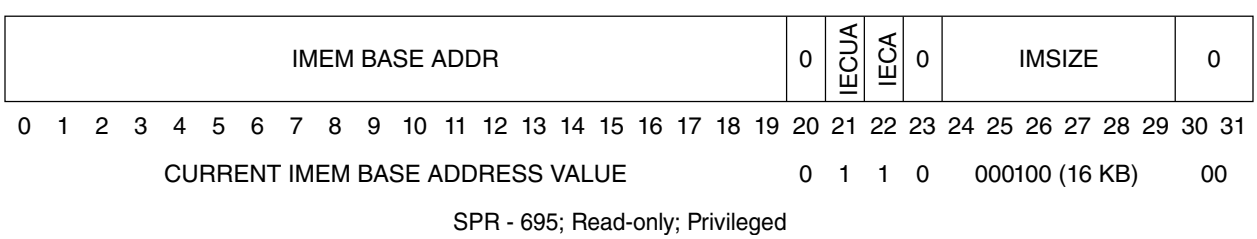


Figure 14-28. IMEM Configuration Register 0 (IMEMCFG0)

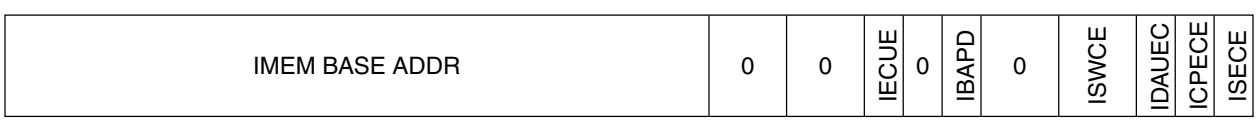
The IMEMCFG0 fields are described in the following table.

Table 14-49. IMEMCFG0 field descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0's when the IMEM size is > 4 KB
20	—	Reserved
21	IECUA	IMEM Error Correction Update Available IECUA indicates an error scrubbing function for the IMEM is available. 0 Error Correction Update is not available 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22	IECA	IMEM Error Correction Available IECA indicates an error correction function for the IMEM is available. 0 Error Correction is not available 1 Error Correction is available
23	—	Reserved
24:29	IMEM Size	IMSIZE - IMEM Size 000100 The size of the IMEM is 16 KB
30:31	—	Reserved

14.10.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.



DCR - 497; Read/Write; Reset[1] - 22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]; Privileged

Figure 14-29. IMEM Control Register 0 (IMEMCTL0)

Note

[1] Reset by an internal Power-On-Reset signal or by an internal destructive reset signal. Unaffected by **p_reset_b**.

Table 14-50. IMEMCTL0 field descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the I-Cache for the memory space to be occupied by the IMEM.
18:19	—	Reserved for Base Address extension
20:21	—	Reserved
22	IECUE	IMEM Error Correction Update Enable IECUE can be used to provide an error scrubbing function. This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[0]** input, an internally hard coded signal. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read cycles will cause the IMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (ICPECE for CPU accesses, ISECE for slave port accesses). IECUE can be used in conjunction with ICPECE and ISECE to provide an error scrubbing function.
23	—	Reserved
24	IBAPD	IMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses. 0 The external IMEM base address port is used to define the base address of the IMEM for CPU accesses. 1 The external IMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses.
25:26	—	Reserved
27:28	ISWCE	IMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the values of the **p_imem_rstcfg[1:2]** inputs, which are internally hard coded signals.
29	IDAUEC	IMEM Disable Address Use in Error Check

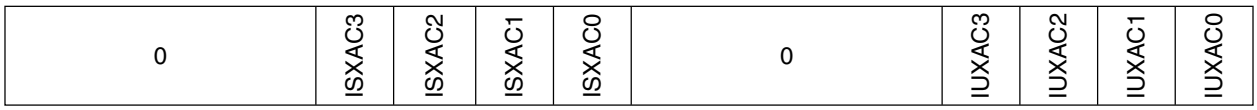
Table continues on the next page...

Table 14-50. IMEMCTL0 field descriptions (continued)

Bits	Name	Description
		This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[3]** input, which is an internally hard coded signal.
30	ICPECE	IMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[4]** input, which is an internally hard coded signal.
31	ISECE	IMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[5]** input, which is an internally hard coded signal.

14.10.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.



DCR - 499; Read/Write; Reset [1] - 32‘b0; Privileged

Figure 14-30. IMEM Control Register 1 (IMEMCTL1)

Note

[1] Reset by **m_por** and **p_reset_b**.

Table 14-51. IMEMCTL1 field descriptions

Bits	Name	Description
0:7	—	Reserved
8:9	ISXAC3	IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2'b11
10:11	ISXAC2	IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2'b10
12:13	ISXAC1	IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 14-51. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2'b01
14:15	ISXAC0	IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2'b00
16:23	—	Reserved
24:25	IUXAC3	IMEM User Instruction Fetch Access Control for Quadrant 3 00 User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2'b11
26:27	IUXAC2	IMEM User Instruction Fetch Access Control for Quadrant 2 00 User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2'b10
28:29	IUXAC1	MEM User Instruction Fetch Access Control for Quadrant 1 00 User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 14-51. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2'b01
30:31	IUXAC0	IMEM User Instruction Fetch Access Control for Quadrant 0 00 = User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 = User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 = User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 = Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2'b00

14.11 End-to-End ECC Support

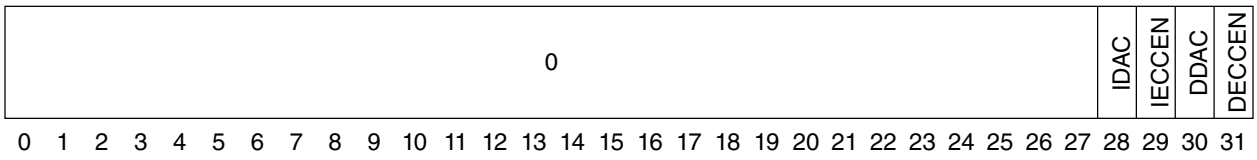
This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

14.11.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

14.11.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.



DCR - 510; Read/Write; Reset - 0x5 || p_e2e_rstcfg[0:3]; Privileged

Figure 14-31. e2eECC Control Register 0 (E2ECTL0)

Table 14-52. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

14.11.1.2 End-to-End ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/ correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus.

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16 17	18 19	20 21 22 23	24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 14-32. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 14-53. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection:

Table 14-53. E2EECSR0 field descriptions

Bits	Name	Description
		0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 15
Core description

15.1 Overview of the e200z710n3 core

The e200z710n3 is a single-issue 32-bit *Power ISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z710n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *Power ISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The e200z710n3 core integrates an integer execution unit, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining three read and two write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z710n3 core contains a16 KB Instruction Cache, and a 4 KB Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

15.2 Register model

This section describes the registers implemented in the e200z710n3 core. The *Power ISA 2.06* architecture defines register-to-register operations for all computational instructions.

e200z710n3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

Figure 15-1 and Figure 15-2 show the complete e200z710n3 register set, all of which are accessible in supervisor mode. Figure 15-3 and Figure 15-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

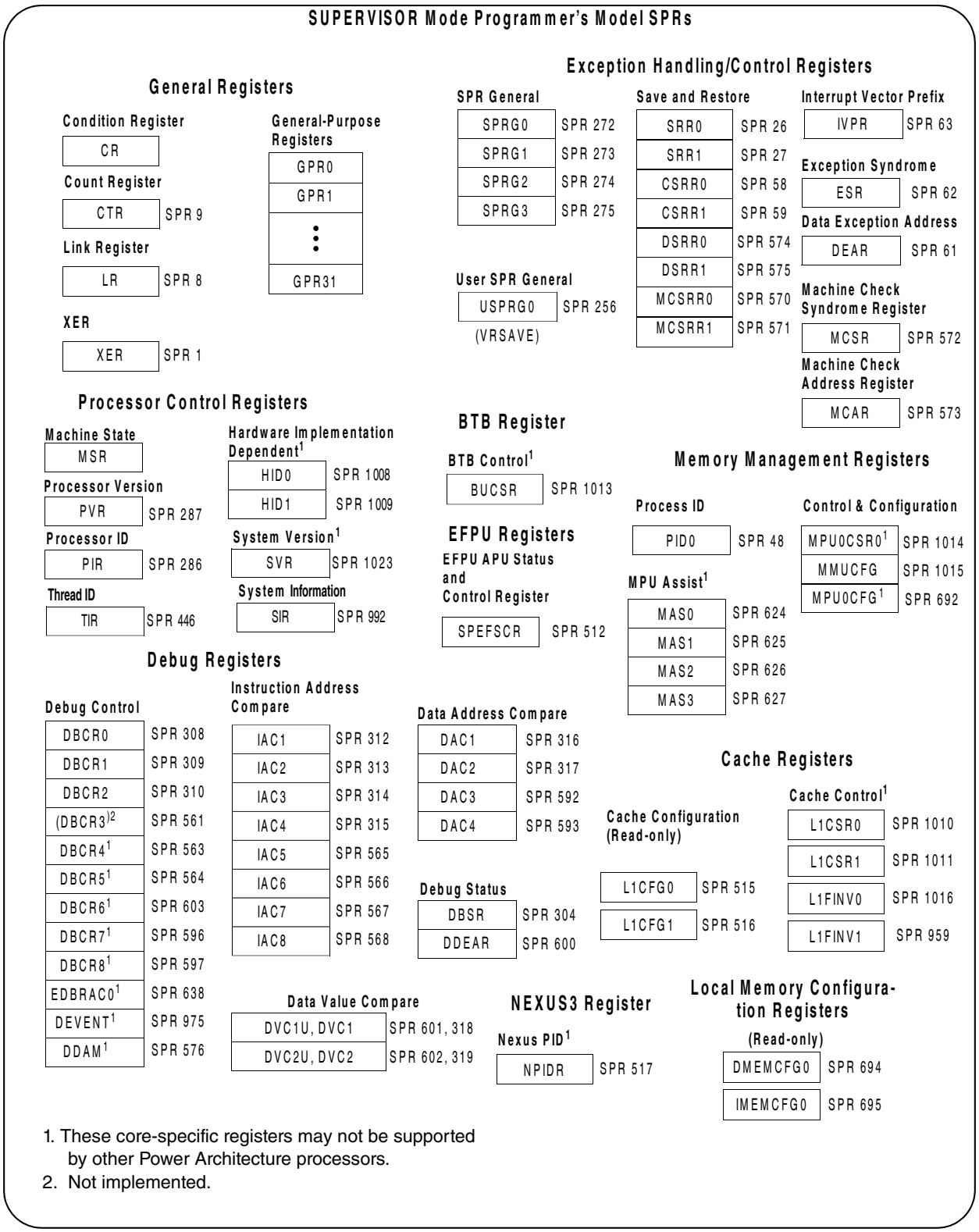


Figure 15-1. e200z710n3 Supervisor Mode Programmer's Model SPRs

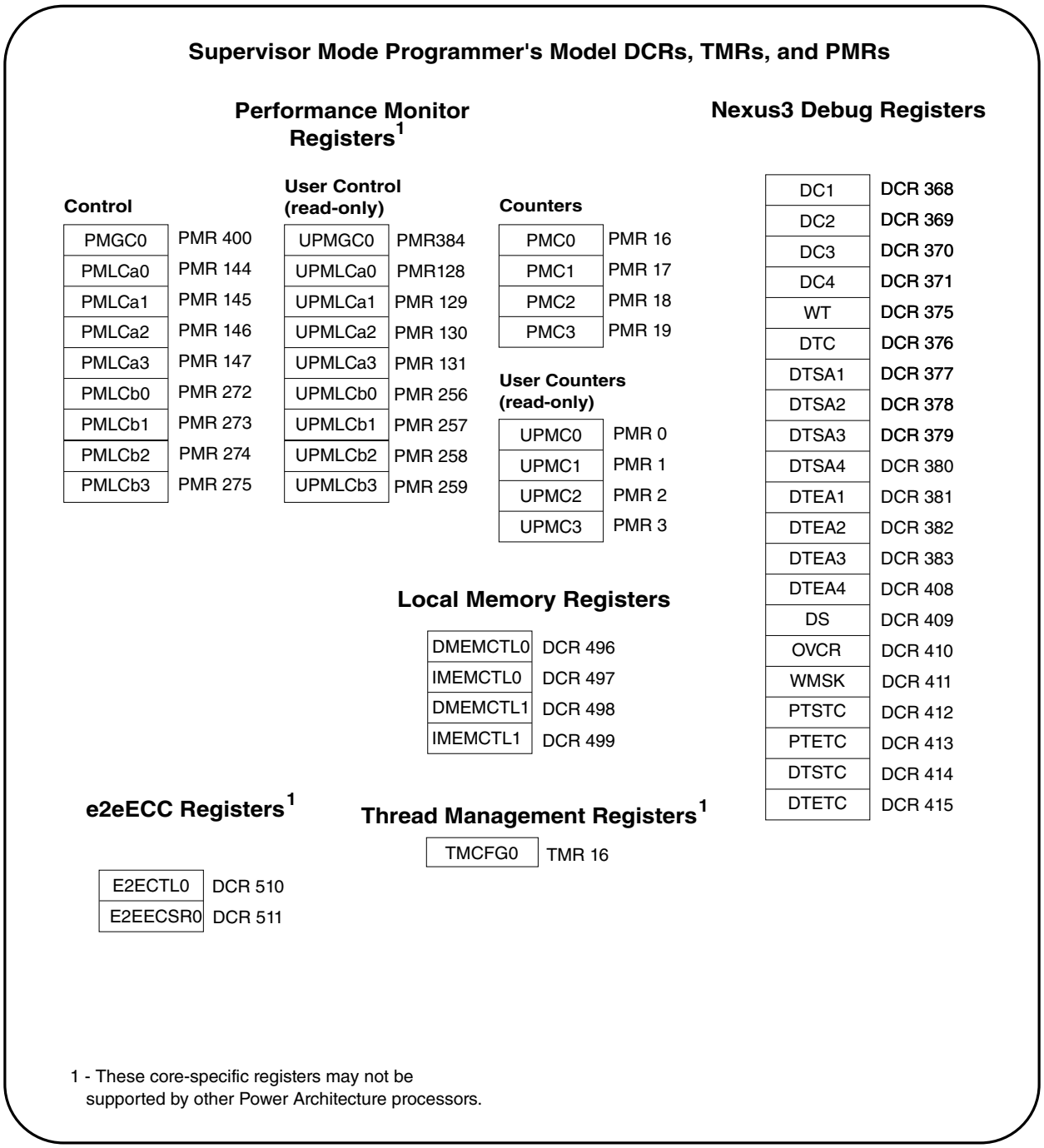


Figure 15-2. e200z710n3 Supervisor Mode Programmer's Model DCRs and PMRs

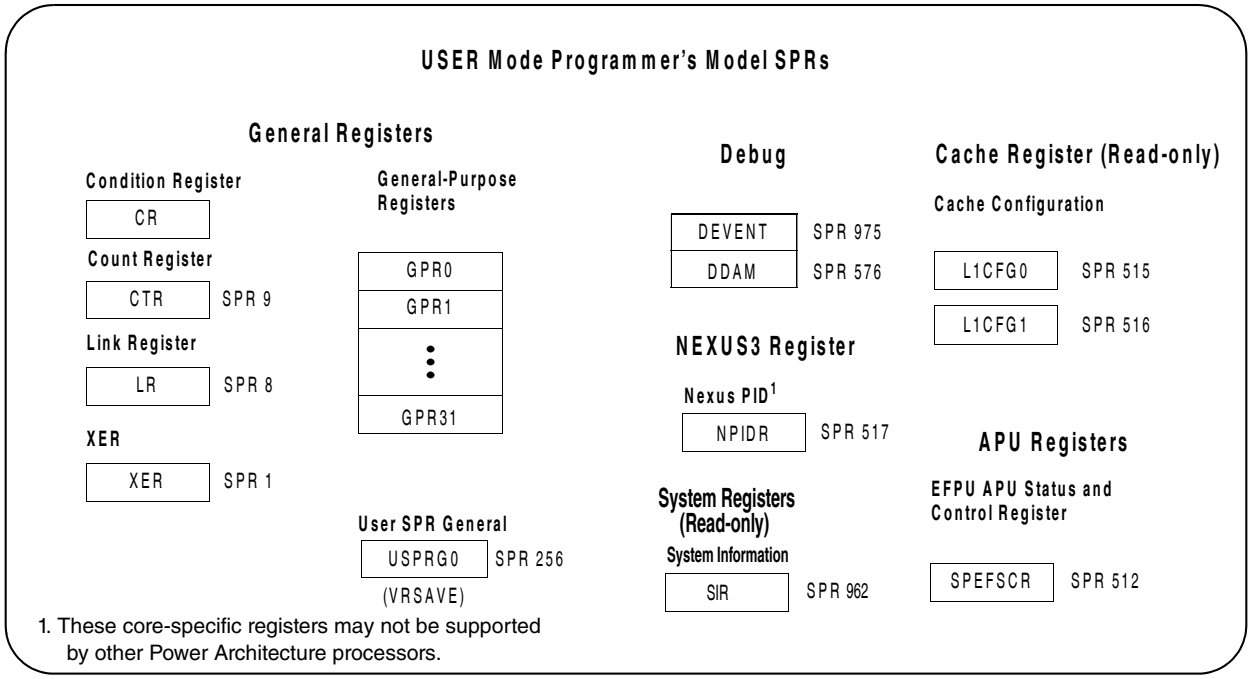


Figure 15-3. e200z710n3 User Mode Programmer's Model SPRs

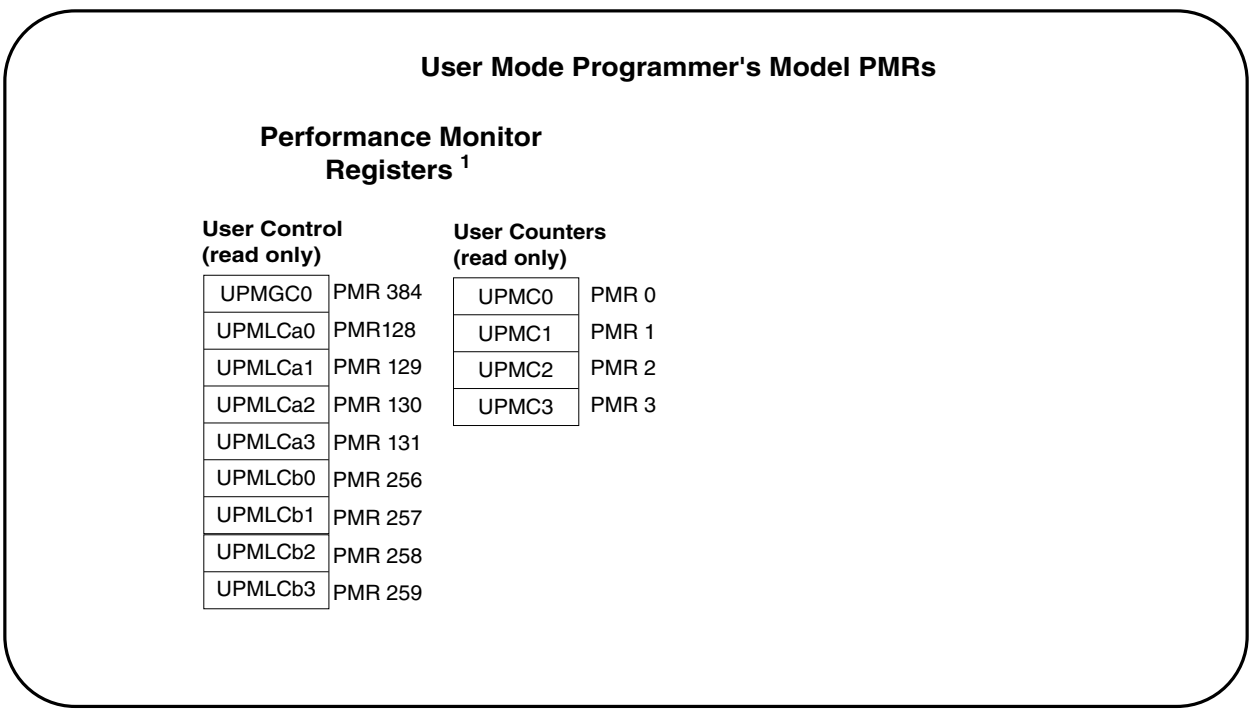
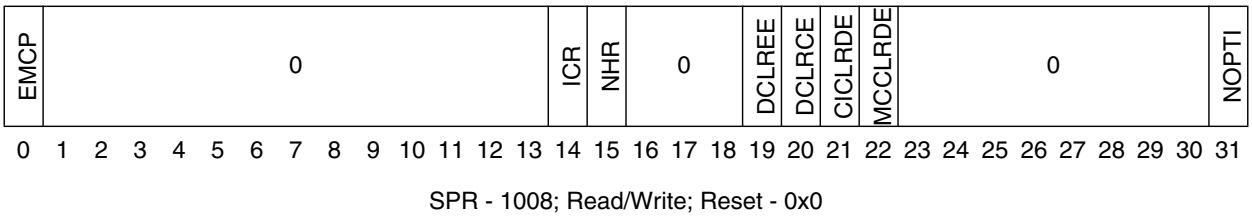


Figure 15-4. e200z710n3 User Mode Programmer's Model PMRs

15.2.1 Hardware Implementation Dependent Register 0 (HID0)

The HID0 register is a core implementation dependent register used for various configuration and control functions. It is shown in the following figure.



SPR - 1008; Read/Write; Reset - 0x0

Figure 15-5. Hardware Implementation Dependent Register 0 (HID0)

The HID0 fields are defined in the following table.

Table 15-1. Hardware Implementation-Dependent Register 0

Bits	Name	Description
0 [32]	EMCP	Enable machine check pin (p_mcp_b) 0 p_mcp_b pin is disabled. 1 p_mcp_b pin is enabled. Asserting p_mcp_b causes a machine check interrupt to be reported.
1:13 [33:45]	-	Reserved
14 [46]	ICR	Interrupt Inputs Clear Reservation 0 External Input, Critical Input, and Non-Maskable Interrupts do not affect reservation status 1 External Input, Critical Input, and Non-Maskable Interrupts clear an outstanding reservation
15 [47]	NHR	Not hardware reset 0 indicates to a reset exception handler that a reset occurred if software had previously set this bit 1 indicates to a reset exception handler that no reset occurred if software had previously set this bit Provided for software use. Set anytime by software, cleared by reset.
16:18 [48:50]	-	Reserved
19 [51]	DCLREE	Debug Interrupt Clears MSR_{EE} 0 MSR_{EE} unaffected by Debug Interrupt 1 MSR_{EE} cleared by Debug Interrupt This bit controls whether Debug interrupts force External Input interrupts to be disabled, or whether they remain unaffected.
20 [52]	DCLRCE	Debug Interrupt Clears MSR_{CE} 0 MSR_{CE} unaffected by Debug Interrupt 1 MSR_{CE} cleared by Debug Interrupt This bit controls whether Debug interrupts force Critical interrupts to be disabled, or whether they remain unaffected.
21 [53]	CICLRDE	Critical Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Critical class interrupt

Table continues on the next page...

Table 15-1. Hardware Implementation-Dependent Register 0 (continued)

Bits	Name	Description
		1 MSR_{DE} cleared by Critical class interrupt This bit controls whether Critical interrupts force Debug interrupts to be disabled, or whether they remain unaffected. Machine Check interrupts have a separate control bit. Note that if Critical Interrupt Debug events are enabled ($DBCR0_{CIRPT}$ set), and MSR_{DE} is set at the time of a Critical Input Critical interrupt, a debug event will be generated after the Critical Interrupt Handler has been fetched, and the Debug handler will be executed first. In this case, $DSRR0_{DE}$ will have been cleared, such that after returning from the debug handler, the Critical interrupt handler will not be run with MSR_{DE} enabled.
22 [54]	MCCLRDE	Machine Check Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Machine Check interrupt 1 MSR_{DE} cleared by Machine Check interrupt This bit controls whether Machine Check interrupts force Debug interrupts to be disabled, or whether they remain unaffected.
23:30 [58:62]	-	Reserved
31 [63]	NOPTI	No-op Touch Instructions 0 icbt, dcbt, dcbtst instructions operate normally 1 icbt, dcbt, dcbtst instructions are no-oped This bit only affects the icbt, dcbt, and dcbtst instructions.

15.2.2 Hardware Implementation Dependent Register 1 (HID1)

The HID1 register is used for bus configuration and system control. It is shown in the following figure.

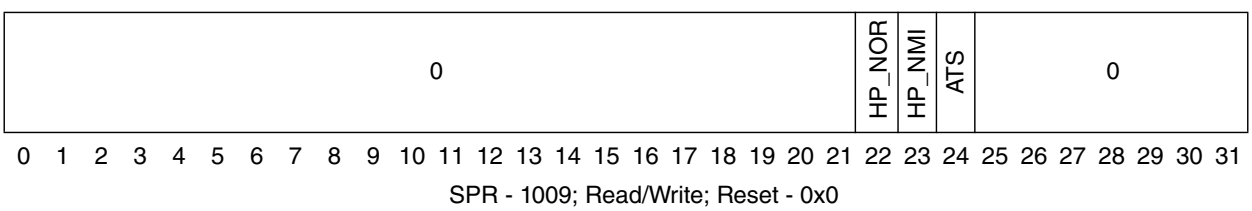


Figure 15-6. Hardware Implementation Dependent Register 1 (HID1)

The HID1 fields are defined in the following table.

Table 15-2. Hardware Implementation-Dependent Register 1

Bits	Name	Description
0:21 [32:53]	-	Reserved

Table continues on the next page...

Table 15-2. Hardware Implementation-Dependent Register 1 (continued)

Bits	Name	Description
22 [54]	HP_NOR	High priority elevation for normal and external interrupts 0 - High priority elevation for normal and external interrupts is disabled 1 - High priority elevation for normal and external interrupts is enabled The HP_NOR bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided. These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
23 [55]	HP_NMI	High priority elevation for NMI and critical interrupts 0 - High priority elevation for NMI and critical interrupts is disabled 1 - High priority elevation for NMI and critical interrupts is enabled The HP_NMI bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
24 [56]	ATS	Atomic status (read-only) Indicates state of the reservation bit in the load/store unit.
25:31 [57:63]	-	Reserved

15.3 Single-issue operation

The instruction issue unit attempts to issue an instruction to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. As many as two results can be simultaneously written.

15.4 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in Table 15-3. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during execution.

Timing for EFPU instructions is detailed in Table 15-6.

Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 15-3. Instruction class cycle counts

Class of Instructions	Latency	Throughput	Special notes
integer: **add**, **sub**, **shift**, **rotate**, **logical**, **cntlzw**	1	1	—
integer: **compare**	1	1	—
Branch	6/4/1	6/4/1	Correct branch lookahead allows single-cycle execution. Worst-case mispredicted branch is 6 cycles.
multiply (saturating/non-saturating)	2/3	1	Result data is available after 2 cycles for non-saturating forms, and 3 cycles for saturating forms, and record form conditions are available after 3 cycles for both forms.
divide	4–15	4–15	Data-dependent timing
CR logical	1	1	—
loads (non-multiple)	3	1	—
load multiple	4 + *n*/2 (max)	4 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
stores (non-multiple)	3	1	—
store multiple	4 + *n*/2 (max)	4 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
mtmsr, **wrtee**, **wrteei**	6&	6&	—
mcrf	1	1	—
mfspr, **mtspr**	4&	4&	Applies to Debug SPRs, optional unit SPRs
mfspr, **mfmsr**	1	1	Applies to certain limited internal, non Debug SPRs
mfcr, **mtcr**	1	1	—
se_rfi, **se_rfci**, **se_rfdi**, **se_rfmci**	6	—	—
se_sc, **e_sc**	4	—	—
e_tw	4	—	Trap taken timing

Detailed timing for each instruction mnemonic along with serialization requirements is shown in Table 15-4 and Table 15-5.

Table 15-4. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	6/4/1	—
se_bclri	1	—
se_bctr	6/5/3/1	—
se_bctrl	6/5/3/1	—
se_bgeni	1	—
se_bl	6/4/1	—
se_blr	6/5/3/1	—
se_blrl	6/5/3/1	—
se_bmaski	1	—
se_b	6/4/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—
se_cmpl	1	—
se_cmpli	1	—
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	3	—
se_lhz	3[2]	—
se_li	1	—
se_lwz	3[2]	—
se_mfar	1	—

Table continues on the next page...

Table 15-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_mfctr	1	—
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2/3	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	6	Refetch
se_rfdi	6	Refetch
se_rfi	6	Refetch
se_rfmci	6	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	3	—
se_sth	3[2]	—
se_stw	3[2]	—
se_sub	1	—
se_subf	1	—
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 15-5. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]	1	—
adde[.]	1	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
addeo[.]	1	—
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	6/4/1	—
e_bc	6/4/1	—
e_bcl	6/4/1	—
e_bl	6/4/1	—
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—
cmpl	1	—
e_cmpl16i	1	—
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
dcbst	1	—
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	4–15[1]	—
divwo[.]	4–15[1]	—
divwu[.]	4–15[1]	—
divwuo[.]	4–15[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	3	—
e_lbz	3	—
e_lbzu	3	—
lbzux	3	—
lbzx	3	—
e_lha	3[2]	—
lharx	3	—
e_lhau	3[2]	—
lhaux	3[2]	—
lhax	3[2]	—
lhbrx	3[2]	—
e_lhz	3[2]	—
e_lhzu	3[2]	—
lhzux	3[2]	—
lhzx	3[2]	—
e_li	1	—
e_lis	1	—
e_lmw	4 + (*n*/2)	—
lwarx	3	—
lwbrx	3[2]	—
e_lwz	3[2]	—
e_lwzu	3[2]	—
lwzux	3[2]	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
lwzx	3[2]	—
mbar	1[3]	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	4[3]	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	6[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	1	None
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2/3	—
mulhwu[.]	2/3	—
e_mull2i	2/3	—
e_mulli	2/3	—
mullw[.]	2/3	—
mullwo[.]	2/3	—
nand[.]	1	—
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	4	Refetch

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
slw[.]	1	—
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	3	—
stbcx.	3	—
e_stbu	3	—
stbux	3	—
stbx	3	—
e_sth	3[2]	—
sthbrx	3[2]	—
sthcx.	3	—
e_sthu	3[2]	—
sthux	3[2]	—
sthx	3[2]	—
e_stmw	4 + (*n*/2)	—
e_stw	3[2]	—
stwbrx	3[2]	—
stwcx.	3	—
e_stwu	3[2]	—
stwux	3[2]	—
stwx	3[2]	—
subf[.]	1	—
subfc[.]	1	—
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	6	Completion
wrteei	6	Completion

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
xor[.]	1	—
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

Instruction timing for EFPU single-precision scalar floating-point instructions is shown in Table 15-6. The table is sorted by opcode.

Table 15-6. EFPU single-precision scalar floating-point instruction timing

Instruction	Latency	Throughput	Comments
efsabs	4	1	—
efsadd	4	1	—
efscfh	4	1	—
efscfsf	4	1	—
efscfsi	4	1	—
efscfuf	4	1	—
efscfui	4	1	—
efscmpeq	4	1	—
efscmpgt	4	1	—
efscmplt	4	1	—
efscth	4	1	—
efsctsf	4	1	—
efsctsi	4	1	—
efsctsiz	4	1	—
efsctuf	4	1	—
efsctui	4	1	—
efsctuiz	4	1	—
efsdiv	13	13	Blocking, no execution overlap with next instruction
efsmadd	4	1[1]	Destination also used as source
efsmsub	4	1[1]	Destination also used as source
efsmax	4	1	—
efsmin	4	1	—
efsmul	4	1	—
efsnabs	4	1	—
efsneg	4	1	—
efsnmadd	4	1[1]	Destination also used as source
efsnmsub	4	1[1]	Destination also used as source
efssqrt	15	15	Blocking, no overlap with next instruction

Table continues on the next page...

Table 15-6. EFPU single-precision scalar floating-point instruction timing (continued)

Instruction	Latency	Throughput	Comments
efssub	4	1	—
efststeq	4	1	—
efststgt	4	1	—
efststlt	4	1	—

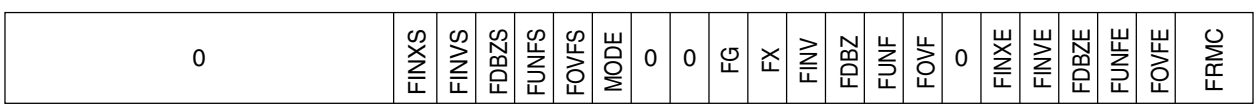
1. Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.

15.5 Reservation instructions and cache interactions

If the CPU supports reservation instruction functionality, the CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

15.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in the following figure.



SPR - 512; Read/Write; Reset - 0x0

Figure 15-7. EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 15-7. EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved

Table continues on the next page...

Table 15-7. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z710n3 Supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by e200 core)
16 (48)	—	Reserved
17 (49)	—	Reserved
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21	FDBZ	Embedded Floating-point Divide by Zero

Table continues on the next page...

Table 15-7. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
(53)		The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22	FUNF	Embedded Floating-point Underflow
(54)		The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23	FOVF	Embedded Floating-point Overflow
(55)		The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24	—	Reserved
(56)		
25	FINXE	Embedded Floating-point Inexact Exception Enable
(57)		If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG l FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable
(58)		If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27	FDBZE	Embedded Floating-point Divide by Zero Exception Enable
(59)		If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28	FUNFE	Embedded Floating-point Underflow Exception Enable
(60)		If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29	FOVFE	Embedded Floating-point Overflow Exception Enable
(61)		If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31	FRMC	Embedded Floating-point Rounding Mode Control
(62:63)		00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity

Table 15-7. EFPU Status and Control Register field descriptions

Bits	Name	Description
		11 Round toward -Infinity

15.7 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

15.7.1 Cache overview

The e200z710n3 processor supports a16 KB 2-way set-associative instruction and 4KB 2-way set-associative data cache with a 32-byte line size. The caches improves system performance by providing low-latency data to the e200z710n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z710n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

15.7.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR0 register is shown in the following figure.

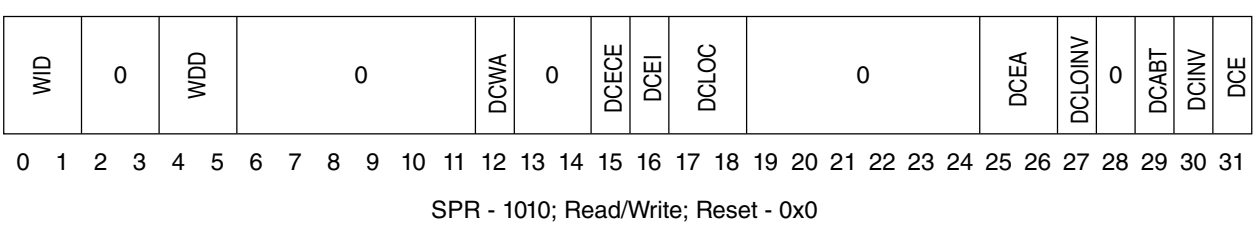


Figure 15-8. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 15-8. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:3	—	Reserved[1]
4:5	WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills. 1 The corresponding way in the data cache is not available for replacement by data miss line fills.
8:11	—	Reserved[1]
12	DCWA	Data Cache Write Allocation Policy This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
13:14	—	Reserved[1]
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.
17:18	DCLOC	Data Cache Lockout Control **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled.

Table continues on the next page...

Table 15-8. L1CSR0 field descriptions (continued)

Bits	Name	Description
		Note: When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table continues on the next page...

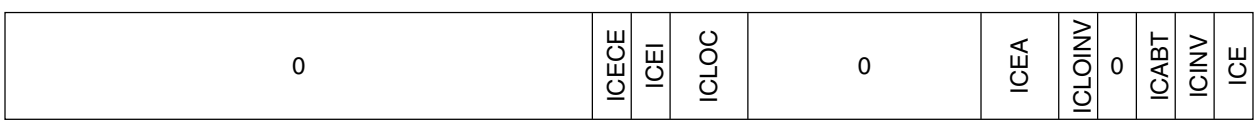
Table 15-8. L1CSR0 field descriptions (continued)

Bits	Name	Description
30	DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31	DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses.Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

15.7.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR1 register is shown in the following figure.



SPR - 1011; Read/Write; Reset - 0x0

Figure 15-9. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 15-9. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved[1]
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable

Table continues on the next page...

Table 15-9. L1CSR1 field descriptions (continued)

Bits	Name	Description
		ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits.
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation

Table continues on the next page...

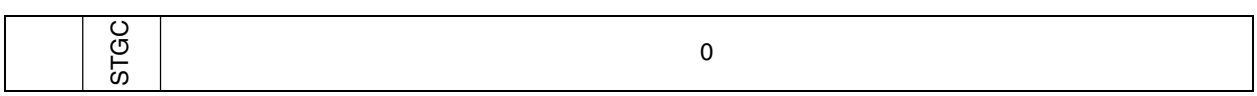
Table 15-9. L1CSR1 field descriptions (continued)

Bits	Name	Description
		operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 258 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

15.7.4 L1 Cache Control and Status Register 2 (L1CSR2)

The L1 Cache Control and Status Register 2 (L1CSR2) is a 32-bit register used for extended cache control. The L1CSR2 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR2 is 606 in decimal, and is shown in the following figure.



SPR - 606; Read/Write; Reset - 0x0

Figure 15-10. L1 Cache Control and Status Register 2 (L1CSR2)

The L1CSR2 bits are described in the following table.

Table 15-10. L1CSR2 field descriptions

Bits	Name	Description
0:1	—	Reserved
2:3	STGC	Store Gather Control 00 - Default Operation - Implementation defined. for e200 store gathering is disabled. 01 - Store Gathering is enabled. No constraints on alignment or requirement to be contiguous. External bus write accesses for gathered stores may have non-contiguous byte strobes asserted. 10 - Store Gathering is enabled. Gathered stores are required to be contiguous once gathered. Cache hit data from a store access lookup may be used to provide adjacent store data to increase gathering opportunities. External bus write accesses for gathered stores will not have non-contiguous byte strobes asserted. 11 - No Store Gathering is Performed
4:31	—	Reserved

15.7.5 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z710n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The L1CFG0 register is shown in the following figure.

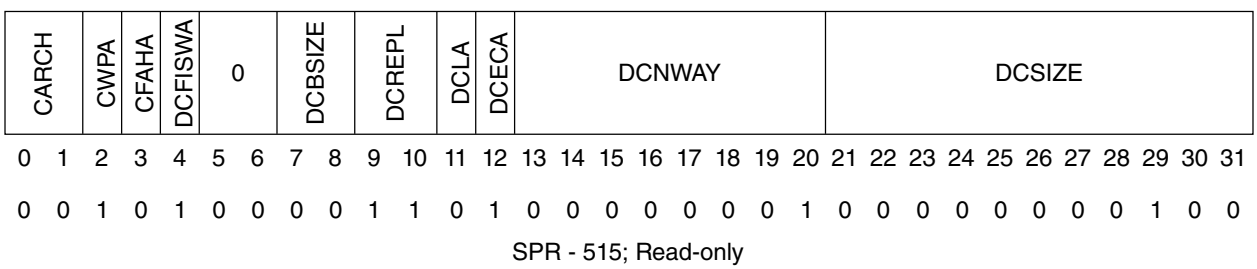


Figure 15-11. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in the following table.

Table 15-11. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available

Table continues on the next page...

Table 15-11. L1CFG0 field descriptions (continued)

Bits	Name	Description
		1 The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 The size of the data cache is 4 KB

15.7.6 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z710n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The L1CFG1 register is shown in the following figure.

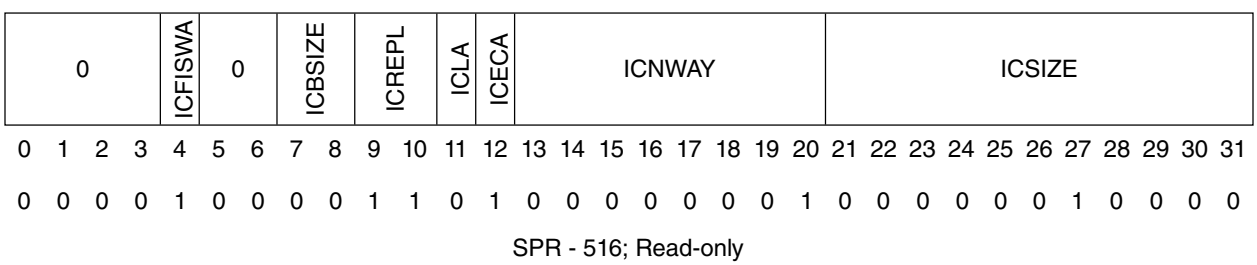


Figure 15-12. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 15-12. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x010 The size of the instruction cache is 16 KB

15.7.7 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

15.7.8 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle.

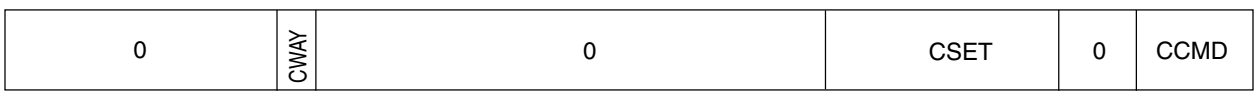
15.7.9 Cache Invalidate by Set and Way

e200z710n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

15.7.9.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in the following figure.



SPR - 1016; Read/Write; Reset - 0x0

Figure 15-13. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in the following table.

Table 15-13. L1FINV0 field descriptions

Bits	Name	Description
0:6	—	Reserved[1] for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8: 20	—	Reserved[1] for set extension
21 :26	CSET	Cache Set Specifies the cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 = The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.7.9.2 L1FINV1

The L1FINV1 register is shown in following figure.

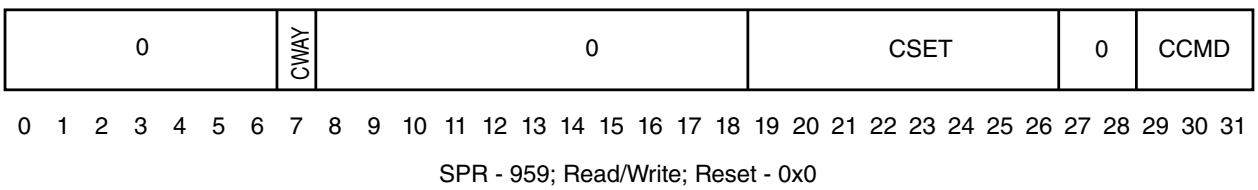


Figure 15-14. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 15-14. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved[1] for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:18	—	Reserved[1] for set extension
19 :26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.7.10 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR0[DCECE] and L1CSR1[ICECE] control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache ($L1CSR\{0,1\}_{[D,I]CECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

15.7.10.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by L1CSR{0,1}$_{[D,I]CLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via L1CSR{0,1}$_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

15.7.11 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If $L1CSR0_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded. Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction or an L1FINV{0,1} invalidation operation.

15.8 Exceptions

Interrupts implemented in e200z710n3 and the exception conditions that cause them are listed in the following table.

Table 15-15. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt

Table continues on the next page...

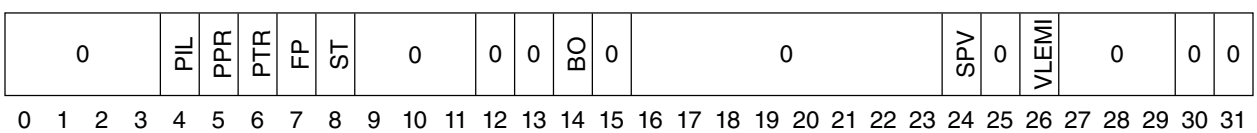
Table 15-15. Exceptions and Conditions (continued)

Interrupt type	Interrupt vector offset value	Causing conditions
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1
EFPU Data Exception	0xA0	Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

15.8.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.



SPR - 62; Read/Write; Reset - 0x0

Figure 15-15. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 15-16. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception	Program

Table continues on the next page...

Table 15-16. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
		For e200z710n3, PIL is used for both illegal and unimplemented instructions.	
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

15.8.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z710n3 MSR is shown in the following figure.

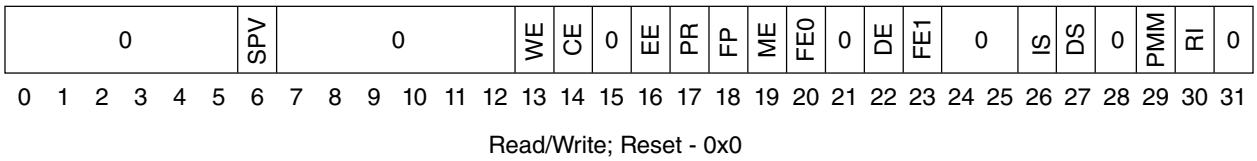


Figure 15-16. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 15-17. MSR field descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/Vector unit is unavailable. The processor cannot execute SP/Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/Vector unit is available. The processor can execute SP/Embedded FP Double or Vector Unit instructions. **NOTE:** The e200z710n3 processor does not support these units in hardware. An Illegal Instruction exception is generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable. **NOTE:** this bit is implementation dependent and is being phased out. It will be removed in a future release. Currently it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	FP	Floating-Point Available 0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **NOTE:** The e200z710n3 processor does not support the PowerISA 2.06 floating point unit in hardware, and an Illegal Instruction exception is generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.

Table continues on the next page...

Table 15-17. MSR field descriptions (continued)

Bit(s)	Name	Description
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used) **NOTE:** The e200z710n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE0 is ignored, but cleared on exceptions.
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used) **NOTE:** The e200z710n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **NOTE:** The e200z710n3 processor does not support Address Spaces, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1. **NOTE:** The e200z710n3 processor does not support Address Spaces, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

15.8.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in the following figure.

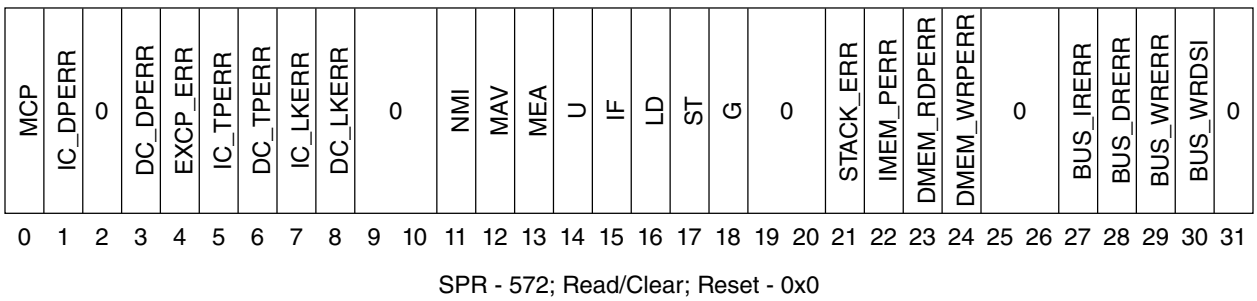


Figure 15-17. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $MSR_{ME} = 1$.

Table 15-18. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error	Async Mchk	—

Table continues on the next page...

Table 15-18. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.		
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[2]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise

Table continues on the next page...

Table 15-18. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
   - "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
2. Note that since no address translation mechanism is present on e200z710n3, the MEA value is always set to '0'.

15.8.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

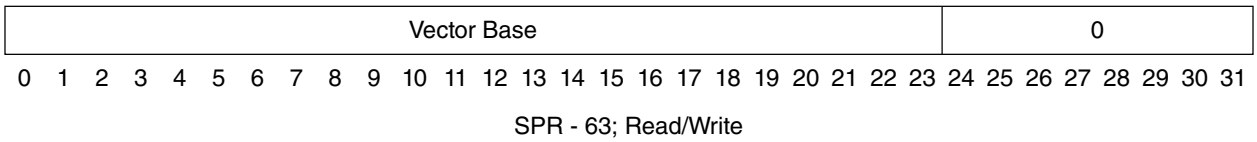


Figure 15-18. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 15-19. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

15.8.5 Interrupt Definitions

15.8.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 15-20. Critical Input Interrupt—register settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0[1]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored)					
	$IVPR_{0:15}$ || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

15.8.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 15-21. Machine Check Interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 15-21. Machine Check Interrupt—register settings (continued)

Register	Setting description					
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

15.8.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 15-22. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

15.8.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 15-23. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

15.8.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 15-24. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					

Table continues on the next page...

Table 15-24. External Input Interrupt—Register Settings (continued)

Register	Setting description
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)

15.8.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

The following table lists register settings when an alignment interrupt is taken.

Table 15-25. Alignment Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

15.8.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception

- Privileged Instruction exception
- Trap exception

The e200z710n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z710n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z710n3 will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *Power ISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 15-26. Program Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			

Table continues on the next page...

Table 15-26. Program Interrupt—register settings (continued)

Register	Setting description	
	Trap:	PTR, VLEMI. All other bits cleared.
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$IVPR_{0:23}$ || 0x60	

15.8.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- $PMLCa x_{CE}$ = 11; that is, for the given counter the overflow condition is enabled.
- $PMC x_{OV}$ = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 15-27. Performance Monitor Interrupt—Register Settings

Register	Setting description					
SRR0/ DSRR0[1]	Set to the effective address of the next instruction to be executed.					
SRR1/ DSRR1[1]	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					

Table continues on the next page...

Table 15-27. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting description
MCSR	Unchanged
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x70

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

15.8.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 15-28. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x80					

15.8.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$ = 1 (internal debug mode) and MSR_{DE} = 1).

The following table lists register settings when a Debug interrupt is taken.

Table 15-29. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			
	External Debug Event:		{DEVT1, DEVT2}			
	Performance Monitor Debug Event:		PMI			
	MPU Debug Event:		MPU			
	and optionally, an Imprecise Debug Event flag		{IDE}			
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x90					

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

15.8.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 15-30. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

15.8.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 15-31. Embedded Floating-point Round Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0

Table continues on the next page...

Table 15-31. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description					
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xB0					

15.8.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 15-32. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

15.9 Memory Protection Unit (MPU)

15.9.1 MPU Overview

The e200z710n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

15.9.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

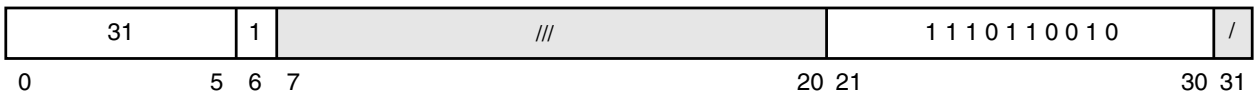
15.9.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

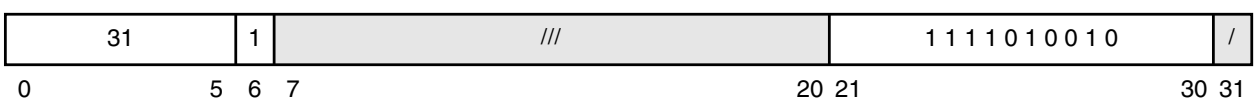
15.9.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

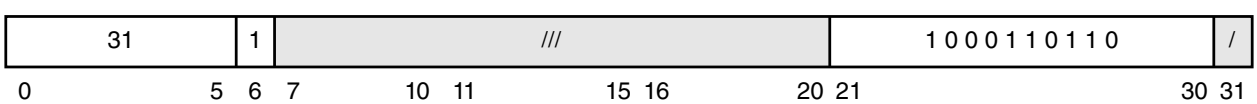
15.9.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the core.

mpusync

MPU Synchronize
mpusync

mpusync



15.9.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z710n3 MPU design. The MMUCFG register is shown in the following figure.

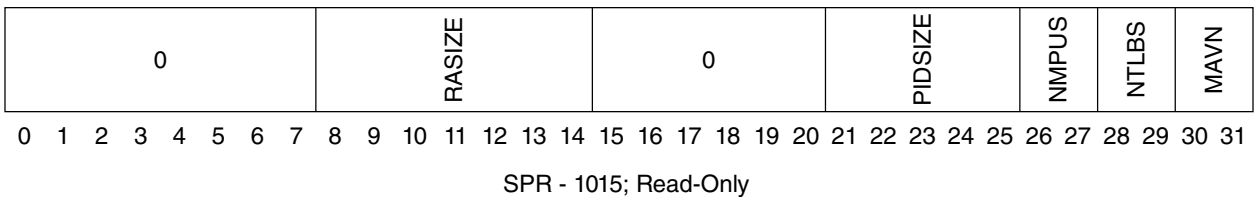


Figure 15-19. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in the following table.

Table 15-33. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

15.9.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z710n3 MPU. The MPU0CFG register is shown in following figure.

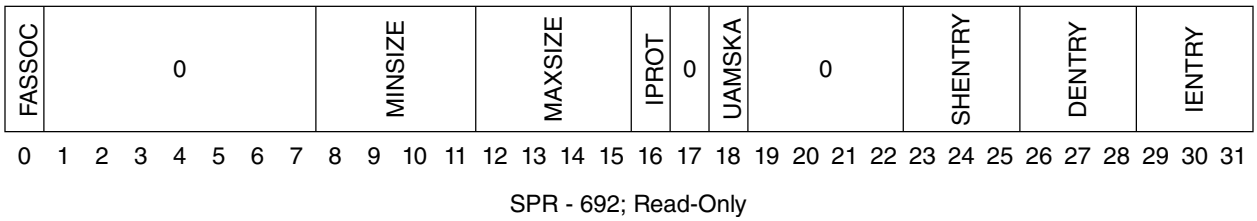


Figure 15-20. MPU0 Configuration Register (MPU0CFG)

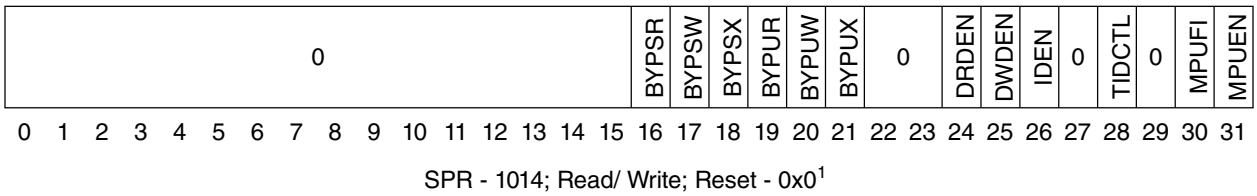
The MPU0CFG bits are described in the following table.

Table 15-34. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

15.9.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in the following figure.



SPR - 1014; Read/ Write; Reset - 0x0[1]

Figure 15-21. MPU0 Control and Status Register 0 (MPU0CSR0)

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM} = 0$, as well as unconditionally by an internal power-on reset signal. If $EDBCR0_{EDM} = 1$, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in the following table.

Table 15-35. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses

Table continues on the next page...

Table 15-35. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYPUW	Bypass MPU protections for User Write accessesThis bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses

Table continues on the next page...

Table 15-35. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation **Note:** Entries that have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

15.9.9 MPU Assist Registers (MAS)

The core uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in Figure 15-22. Fields are defined in Table 15-36.

VALID	IPROT	SEL	0	RO	DEBUG	INST	SHD	0	ESEL	0	UAMSK	UW	SW	UX/UR	SX/SR	IOVR	GOVR[1]	1	1	I	0	G[1]	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 624; Read/ Write; Reset - Unaffected

Figure 15-22. MPU Assist Register 0 (MAS0)

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 15-36. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 = This MPU entry is invalid 1 = This MPU entry is valid
1	IPROT	Invalidation Protect 0 = Entry is not protected from invalidation 1 = Entry is protected from invalidation
2:3	SEL	Selects MPU for access: 00 = Reserved, no access 01 = Reserved, no access 10 = Select MPU 11 = Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 = This MPU entry is writable 1 = This MPU entry is Read-only
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 = This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 = This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 = This MPU entry is used for matching data accesses only 1 = This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table.

Table continues on the next page...

Table 15-36. MAS0—MPU Read/Write and Replacement Control (continued)

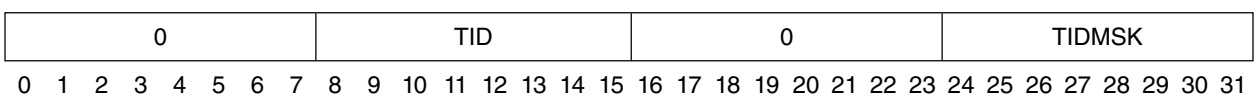
Bit	Name	Comments, or Function when Set
		0 = The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 = The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12 :15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved
17:19	UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 = No upper address bits are masked, address matching uses all 32 address bits for accesses 001 = The most significant access address bit is forced to zero before matching 010 = The two most significant access address bits are forced to zero before matching 011 = The three most significant access address bits are forced to 0 before matching 100 = The four most significant access address bits are forced to 0 before matching 101 = The upper five access address bits are forced to zero before matching 11x = Reserved, do not use
20	UW	User Mode Write PermissionDetermines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No User mode Write permission 1 = User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No Supervisor mode Write permission 1 = Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 = No User mode Execute/ Read permission 1 = User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission

Table continues on the next page...

Table 15-36. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
		Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 = No Supervisor mode Execute/ Read permission 1 = Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute OverrideDetermines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals 0 = No override of I attribute by entry 1 = Entry I bit overrides I attribute
25	GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = No override of G attribute by entry 1 = Entry G bit overrides G attribute
26	—	Reserved
27	—	Reserved
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. When the core memory protection unit (CMPU) is disabled, the CMPU configuration is ignored and this bit is ignored also. 0 = This region is considered cacheable 1 = This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings.This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 = All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in Figure 15-23. Fields are defined in Table 15-37.



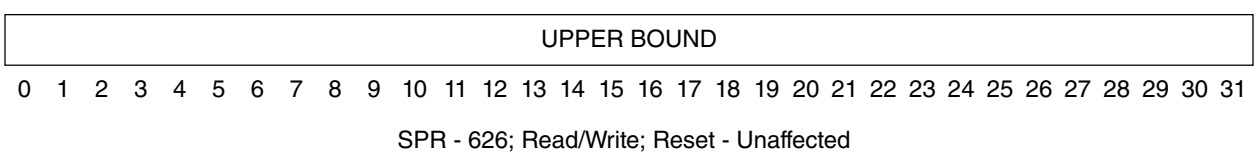
SPR - 625; Read/Write; Reset - Unaffected

Figure 15-23. MPU Assist Register 1 (MAS1)

Table 15-37. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx = TID[0] comparison to PID0[0] not masked 1xxxxxxx = TID[0] comparison to PID0[0] is masked x0xxxxxx = TID[1] comparison to PID0[1] not masked x1xxxxxx = TID[1] comparison to PID0[1] is masked xx0xxxxx = TID[2] comparison to PID0[2] not masked xx1xxxxx = TID[2] comparison to PID0[2] is masked xxx0xxxx = TID[3] comparison to PID0[3] not masked xxx1xxxx = TID[3] comparison to PID0[3] is masked xxxx0xxx = TID[4] comparison to PID0[4] not masked xxxx1xxx = TID[4] comparison to PID0[4] is masked xxxxx0xx = TID[5] comparison to PID0[5] not masked xxxxx1xx = TID[5] comparison to PID0[5] is masked xxxxxx0x = TID[6] comparison to PID0[6] not masked xxxxxx1x = TID[6] comparison to PID0[6] is masked xxxxxxx0 = TID[7] comparison to PID0[7] not masked xxxxxxx1 = TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in Figure 15-24. Fields are defined in Table 15-38.



SPR - 626; Read/Write; Reset - Unaffected

Figure 15-24. MPU Assist Register 2 (MAS2)

Table 15-38. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in Figure 15-25. Fields are defined in Table 15-39.

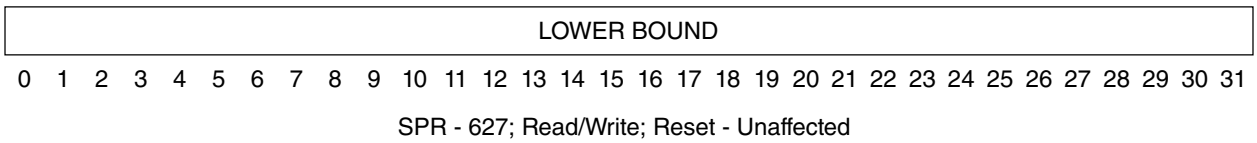


Figure 15-25. MPU Assist Register 3 (MAS3)

Table 15-39. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

15.9.10 MAS Registers Summary

The MAS registers are summarized in the following figure.

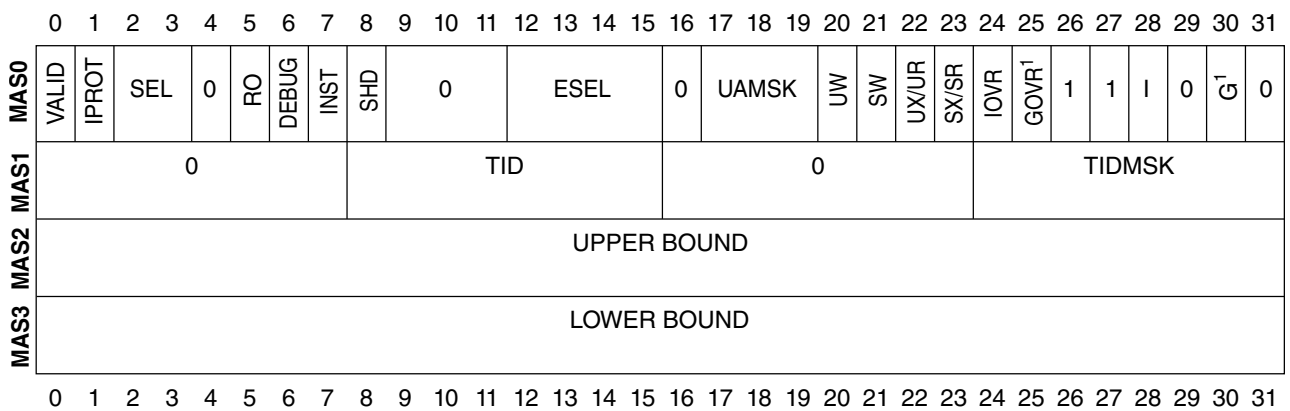


Figure 15-26. MPU Assist Registers Summary

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

15.10 Local memories

15.10.1 Local Instruction and Data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added. This provides low latency access (zero wait-states) similar to an instruction or data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity tightly coupled storage capability.

15.10.2 Local memory control and configuration

DMEM and IMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mfspr** instructions. DMEM and IMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.10.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z710n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in the following figure.

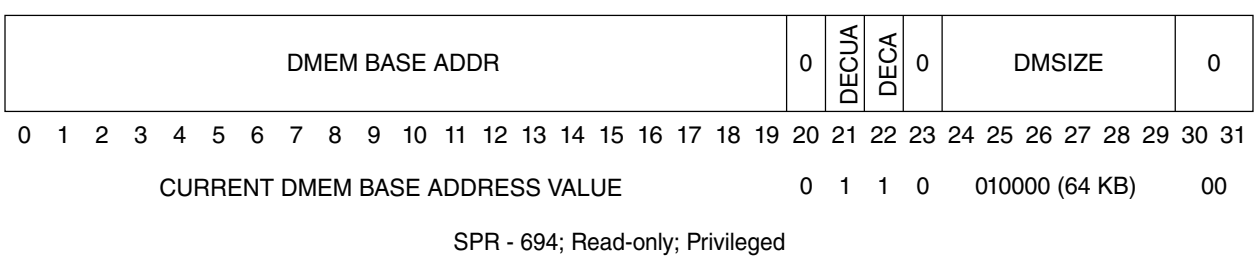


Figure 15-27. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in the following table.

Table 15-40. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port)

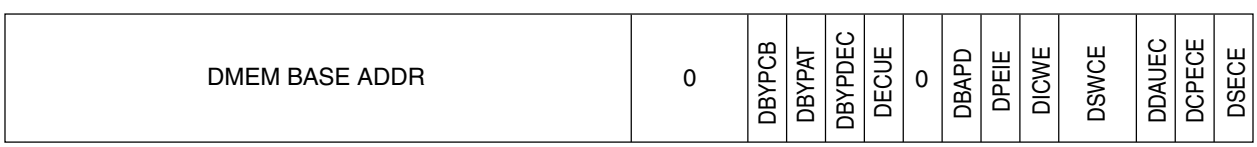
Table continues on the next page...

Table 15-40. DMEMCFG0 field descriptions (continued)

Bits	Name	Description
		This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0s when the DMEM size is > 4 KB
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB.
30:31	—	Reserved

15.10.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.



DCR - 496; Read/Write; Reset[1] - 19'b0 || p_dmem_rstcfg[8,6:7,0] || 4'b0 || p_dmem_rstcfg[1:5]; Privileged

Figure 15-28. DMEM Control Register 0 (DMEMCTL0)

Note

[1] Reset by an internal power-on reset signal or by an internal destructive reset signal. Unaffected by **p_reset_b**.

Table 15-41. DMEMCTL0 field descriptions

Bits	Name	Description
0:15	DMEM BASEADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
16 :18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[8]** input, which is internally hard coded.
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[6]** input, which is internally hard coded.
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.

Table continues on the next page...

Table 15-41. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		Note: This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[7]** input, which is internally hard coded.
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[0]** input which is internally hard coded.
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits.
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected.

Table continues on the next page...

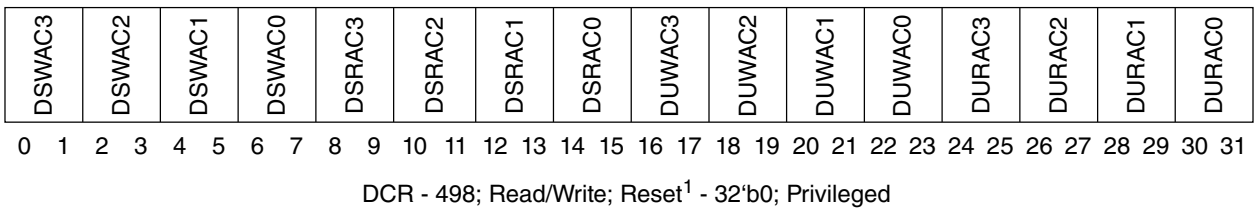
Table 15-41. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the values of the **p_dmem_rstcfg[1:2]** inputs, which are internally hard coded.
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[3]** input, which is an internally hardcoded signal.
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[4]** input, which is internally hard coded.
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[5]** input, which is an internally hard coded signal.

15.10.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally

sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed. All other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB, DBYPATF, DBYPDEC}$.



DCR - 498; Read/Write; Reset[1] - 32‘b0; Privileged

Figure 15-29. DMEM Control Register 1 (DMEMCTL1)

Note

[1] Reset from an internal power-on reset signal or by an internal reset signal.

Table 15-42. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM

Table continues on the next page...

Table 15-42. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 15-42. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 = User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 = User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 = User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 = Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01

Table continues on the next page...

Table 15-42. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM

Table 15-42. DMEMCTL1 field descriptions

Bits	Name	Description
		10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

15.10.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z710n3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in the following figure.

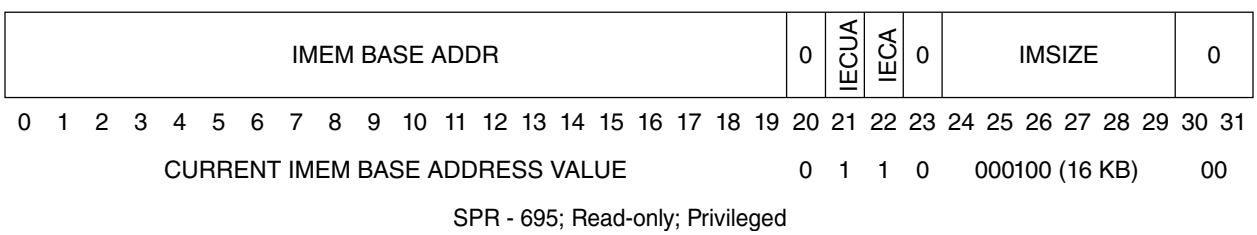


Figure 15-30. IMEM Configuration Register 0 (IMEMCFG0)

The IMEMCFG0 fields are described in the following table.

Table 15-43. IMEMCFG0 field descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0's when the IMEM size is > 4 KB
20	—	Reserved
21	IECUA	IMEM Error Correction Update Available IECUA indicates an error scrubbing function for the IMEM is available. 0 Error Correction Update is not available

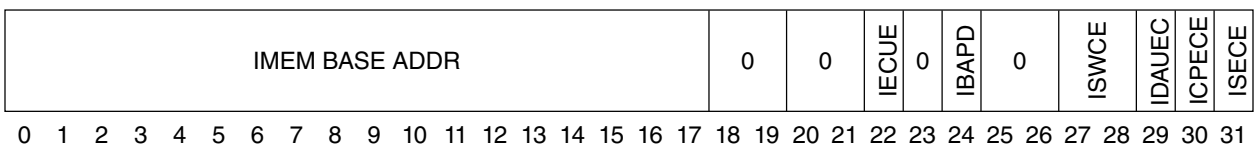
Table continues on the next page...

Table 15-43. IMEMCFG0 field descriptions (continued)

Bits	Name	Description
		1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22	IECA	IMEM Error Correction Available IECA indicates an error correction function for the IMEM is available. 0 Error Correction is not available 1 Error Correction is available
23	—	Reserved
24:29	IMEM Size	IMSIZE - IMEM Size 000100 The size of the IMEM is 16 KB
30:31	—	Reserved

15.10.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.



DCR - 497; Read/Write; Reset[1] - 22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]; Privileged

Figure 15-31. IMEM Control Register 0 (IMEMCTL0)

Note

[1] Reset by an internal Power-On-Reset signal or by an internal destructive reset signal. Unaffected by **p_reset_b**.

Table 15-44. IMEMCTL0 field descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the I-Cache for the memory space to be occupied by the IMEM.
18:19	—	Reserved for Base Address extension
20:21	—	Reserved

Table continues on the next page...

Table 15-44. IMEMCTL0 field descriptions (continued)

Bits	Name	Description
22	IECUE	IMEM Error Correction Update Enable IECUE can be used to provide an error scrubbing function. This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[0]** input, an internally hard coded signal. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read cycles will cause the IMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (ICPECE for CPU accesses, ISECE for slave port accesses). IECUE can be used in conjunction with ICPECE and ISECE to provide an error scrubbing function.
23	—	Reserved
24	IBAPD	IMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses. 0 The external IMEM base address port is used to define the base address of the IMEM for CPU accesses. 1 The external IMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses.
25:26	—	Reserved
27:28	ISWCE	IMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1-, 2-, or 3-byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the values of the **p_imem_rstcfg[1:2]** inputs, which are internally hard coded signals.
29	IDAUEC	IMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[3]** input, which is an internally hard coded signal.
30	ICPECE	IMEM CPU Port ECC Enable (CPU port)

Table continues on the next page...

Table 15-44. IMEMCTL0 field descriptions (continued)

Bits	Name	Description
		0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[4]** input, which is an internally hard coded signal.
31	ISECE	IMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated. **Note:** This field is updated by an internal Power_On_Reset signal or an internal Destructive Reset signal based on the value of the **p_imem_rstcfg[5]** input, which is an internally hard coded signal.

15.10.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.

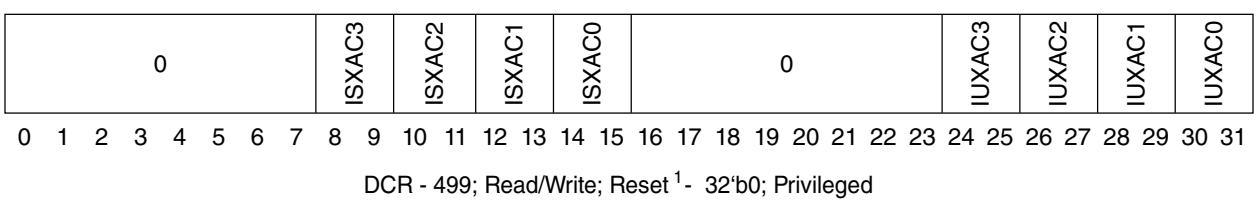


Figure 15-32. IMEM Control Register 1 (IMEMCTL1)

Note

[1] Reset by **m_por** and **p_reset_b**.

Table 15-45. IMEMCTL1 field descriptions

Bits	Name	Description
0:7	—	Reserved
8:9	ISXAC3	IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2'b11
10:11	ISXAC2	IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2'b10
12:13	ISXAC1	IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2'b01
14:15	ISXAC0	IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 15-45. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2'b00
16:23	—	Reserved
24:25	IUXAC3	IMEM User Instruction Fetch Access Control for Quadrant 3 00 User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2'b11
26:27	IUXAC2	IMEM User Instruction Fetch Access Control for Quadrant 2 00 User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2'b10
28:29	IUXAC1	MEM User Instruction Fetch Access Control for Quadrant 1 00 User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2'b01
30:31	IUXAC0	IMEM User Instruction Fetch Access Control for Quadrant 0 00 = User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 = User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 = User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 = Reserved

Table 15-45. IMEMCTL1 field descriptions

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2'b00

15.11 End-to-End ECC Support

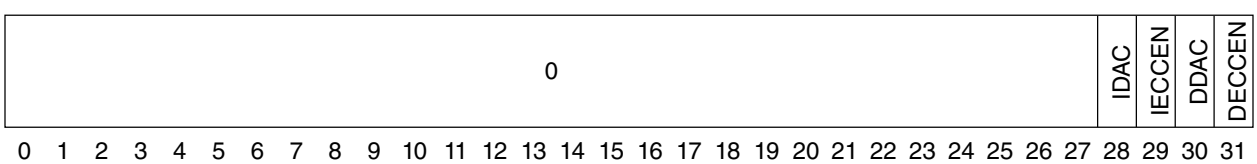
This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

15.11.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

15.11.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.



DCR - 510; Read/Write; Reset - 0x5 || p_e2e_rstcfg[0:3]; Privileged

Figure 15-33. e2eECC Control Register 0 (E2ECTL0)

Table 15-46. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses.

Table continues on the next page...

Table 15-46. E2ECTL0 field descriptions (continued)

Bits	Name	Description
		0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The E2ECTL0$_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

15.11.1.2 End-to-End ECC fault injection by software

Fault injection provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits). For the case of an external read access that receives an ERROR response (HRESP=ERR), the RCHKBIT field is written with all zeros.

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16 17	18 19	20 21 22 23	24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 15-34. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 15-47. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked). If the CPU receives an ERROR response (HRESP=ERR) on an external read access, the RCHKBIT field is written with all zeros.
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further fault injection. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 16
System Integration Unit Lite2 (SIUL2)

16.1 Introduction

16.1.1 Overview

The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports 22 ports with 16 bits of bidirectional, general-purpose input and output signals. It supports ten external interrupts with trigger event configuration. The figure below is the block diagram of SIUL2 and its interfaces to other system components for this device.

The SIUL2 provides the GPIO ports on the device.

- When configured as output, you can write to the internal GPDO register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from the internal GPDI register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking the driven value on the associated I/O pad.

In order to assist in software development, there are different access mechanisms to the GPIO data registers. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

- Access to two 16-bit ports in one access
- Read/Write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access

There are nine external interrupt/DMA inputs to the SIUL2 that map to EIRQ pins on the device. The interrupt/DMA sources can be configured to have a digital filter to reject short glitches on the inputs.

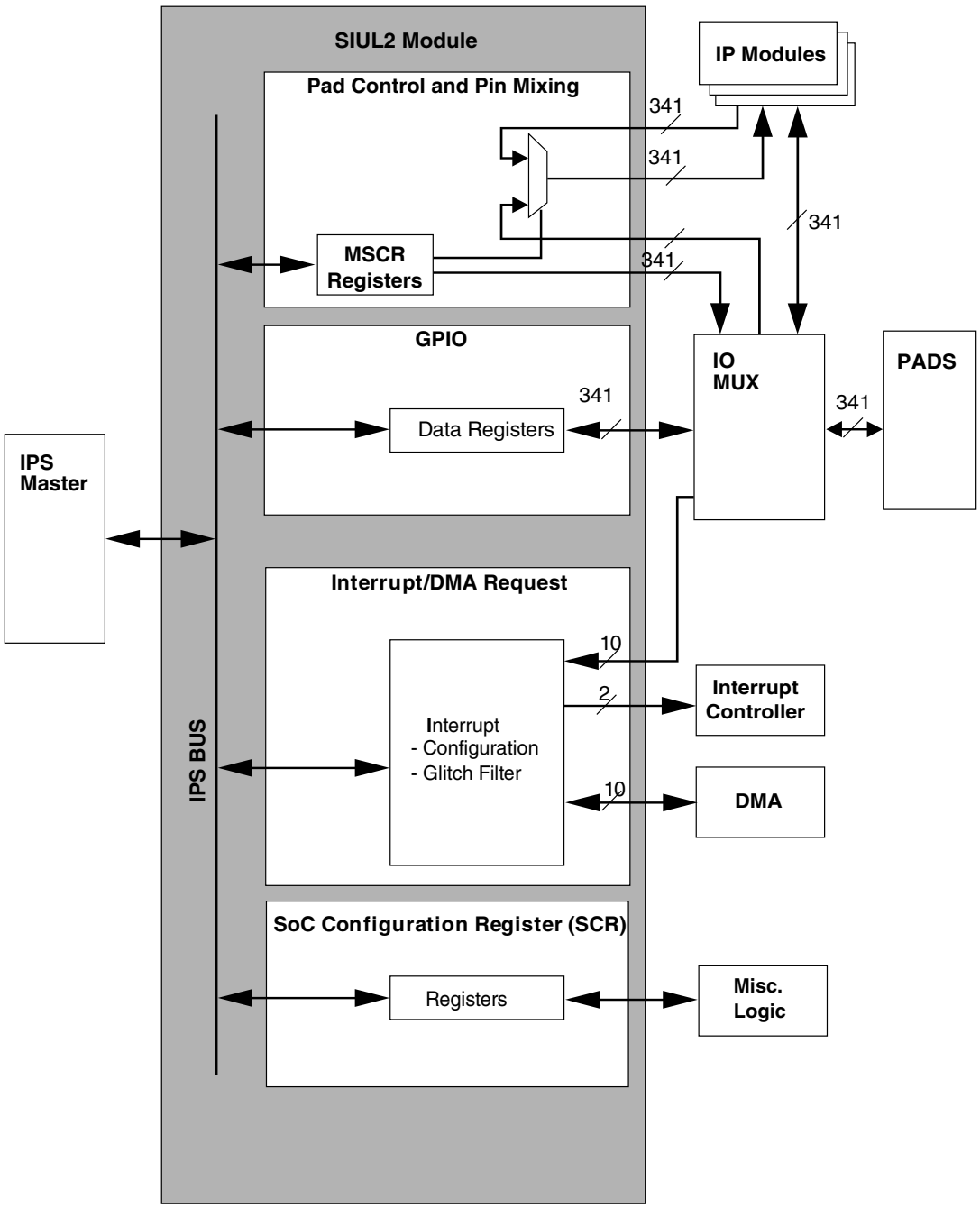


Figure 16-1. SIUL2 block diagram

16.1.2 Features

The SIUL2 supports these distinctive features:

- GPIO ports with data control
  - Drive data to up to 16 independent IO channels
  - Sample data from up to 16 independent IO channels

Read values on two (16-bit) ports at once (PGPDI register), or write values on two ports at once (PGPDO register) with a 32-bit r/w.

External interrupt/DMA request support with:

- Two system interrupt vectors for 10 interrupt sources (10 connected to EIRQ pins for this device) with independent interrupt mask
- 10 programmable digital glitch filters (one for each EIRQ)
- Independent DMA channel for each EIRQ pin
- Edge detection

Additionally the SIUL2 contains the multiplexed signal configuration registers that configure the electrical parameters and settings of all functional pads. These are used to configure the following pad features:

- Drive strength, output impedance, and slew rate control via OERC bits
- INV capability, HYS switch, and IBE switch for source input enable
- Open drain/source output enable
- Internal weak pull control
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration
- TTL, CMOS, and Automotive input levels

16.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0(SIUL2_DIRER0)

- SIUL2 DMA/Interrupt Request Select Register 0(SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0(SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0(SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0(SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 SoC Configuration Register0 (SIUL2_SCR0)
- SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0–SIUL2_MSCR1023)

Details about the protected registers at specific addresses can be found in the Register Protection (REG_PROT) chapter and the Device Configuration chapter.

16.2 Memory map and register description

This section provides a detailed description of all registers accessible in this module.

The following table gives an overview of the module registers implemented.

NOTE

Reserved registers will be read as 0, write will have no effect. If supported and enabled in this MCU, a transfer error will be issued when trying to access completely reserved register space. A 8-bit or 16-bit access to an "only 32-bit supported access register" gives a transfer error. Unimplemented registers accesses give transfer error, i.e. due to register not being present.

NOTE

For SIUL2_MSCR_MUX_512-SIUL2_MSCR_MUX_1023, only 32/16-bit accesses are supported (8-bit writes don't occur and read zeros).

SIUL2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	MCU ID Register #1 (SIUL2_MIDR1)	32	R	See section	16.2.1/736
8	MCU ID Register #2 (SIUL2_MIDR2)	32	R	4800_4D00h	16.2.2/737
10	DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)	32	w1c	0000_0000h	16.2.3/738
18	DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)	32	R/W	0000_0000h	16.2.4/740
20	DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)	32	R/W	0000_0000h	16.2.5/742
28	Interrupt Rising-Edge Event Enable Register0 (SIUL2_IREER0)	32	R/W	0000_0000h	16.2.6/744
30	Interrupt Falling-Edge Event Enable Register0 (SIUL2_IFEER0)	32	R/W	0000_0000h	16.2.7/745
38	Interrupt Filter Enable Register0 (SIUL2_IFER0)	32	R/W	0000_0000h	16.2.8/747
40	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)	32	R/W	0000_0000h	16.2.9/748
44	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR1)	32	R/W	0000_0000h	16.2.9/748
48	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR2)	32	R/W	0000_0000h	16.2.9/748
4C	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR3)	32	R/W	0000_0000h	16.2.9/748
50	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR4)	32	R/W	0000_0000h	16.2.9/748
54	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR5)	32	R/W	0000_0000h	16.2.9/748
58	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR6)	32	R/W	0000_0000h	16.2.9/748
5C	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR7)	32	R/W	0000_0000h	16.2.9/748
60	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR8)	32	R/W	0000_0000h	16.2.9/748
64	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR9)	32	R/W	0000_0000h	16.2.9/748
68	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR10)	32	R/W	0000_0000h	16.2.9/748
6C	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR11)	32	R/W	0000_0000h	16.2.9/748
70	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR12)	32	R/W	0000_0000h	16.2.9/748
74	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR13)	32	R/W	0000_0000h	16.2.9/748
78	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR14)	32	R/W	0000_0000h	16.2.9/748
7C	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR15)	32	R/W	0000_0000h	16.2.9/748
80	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR16)	32	R/W	0000_0000h	16.2.9/748
84	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR17)	32	R/W	0000_0000h	16.2.9/748
88	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR18)	32	R/W	0000_0000h	16.2.9/748
8C	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR19)	32	R/W	0000_0000h	16.2.9/748
90	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR20)	32	R/W	0000_0000h	16.2.9/748

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
94	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR21)	32	R/W	0000_0000h	16.2.9/748
98	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR22)	32	R/W	0000_0000h	16.2.9/748
9C	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR23)	32	R/W	0000_0000h	16.2.9/748
A0	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR24)	32	R/W	0000_0000h	16.2.9/748
A4	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR25)	32	R/W	0000_0000h	16.2.9/748
A8	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR26)	32	R/W	0000_0000h	16.2.9/748
AC	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR27)	32	R/W	0000_0000h	16.2.9/748
B0	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR28)	32	R/W	0000_0000h	16.2.9/748
B4	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR29)	32	R/W	0000_0000h	16.2.9/748
B8	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR30)	32	R/W	0000_0000h	16.2.9/748
BC	Interrupt Filter Maximum Counter Register (SIUL2_IFMCR31)	32	R/W	0000_0000h	16.2.9/748
C0	Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	32	R/W	0000_0000h	16.2.10/ 749
100	SoC Configuration Register0 (SIUL2_SCR0)	32	R/W	0000_0000h	16.2.11/ 750
240	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0)	32	R/W	0009_0000h	16.2.12/ 751
244	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_1)	32	R/W	0009_0000h	16.2.12/ 751
248	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_2)	32	R/W	0009_0000h	16.2.12/ 751
24C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_3)	32	R/W	0009_0000h	16.2.12/ 751
250	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_4)	32	R/W	0009_0000h	16.2.12/ 751
254	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_5)	32	R/W	0009_0000h	16.2.12/ 751
258	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_6)	32	R/W	0009_0000h	16.2.12/ 751
25C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_7)	32	R/W	0009_0000h	16.2.12/ 751
260	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_8)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
264	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_9)	32	R/W	0009_0000h	16.2.12/ 751
268	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_10)	32	R/W	0009_0000h	16.2.12/ 751
26C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_11)	32	R/W	0009_0000h	16.2.12/ 751
270	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_12)	32	R/W	0009_0000h	16.2.12/ 751
274	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_13)	32	R/W	0009_0000h	16.2.12/ 751
278	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_14)	32	R/W	0009_0000h	16.2.12/ 751
27C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_15)	32	R/W	0009_0000h	16.2.12/ 751
280	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_16)	32	R/W	0009_0000h	16.2.12/ 751
284	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_17)	32	R/W	0009_0000h	16.2.12/ 751
288	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_18)	32	R/W	0009_0000h	16.2.12/ 751
28C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_19)	32	R/W	0009_0000h	16.2.12/ 751
290	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_20)	32	R/W	0009_0000h	16.2.12/ 751
294	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_21)	32	R/W	0009_0000h	16.2.12/ 751
298	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_22)	32	R/W	0009_0000h	16.2.12/ 751
29C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_23)	32	R/W	0009_0000h	16.2.12/ 751
2A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_24)	32	R/W	0009_0000h	16.2.12/ 751
2A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_25)	32	R/W	0009_0000h	16.2.12/ 751
2A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_26)	32	R/W	0009_0000h	16.2.12/ 751
2AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_27)	32	R/W	0009_0000h	16.2.12/ 751
2B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_28)	32	R/W	0009_0000h	16.2.12/ 751
2B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_29)	32	R/W	0009_0000h	16.2.12/ 751
2B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_30)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_31)	32	R/W	0009_0000h	16.2.12/ 751
2C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_32)	32	R/W	0009_0000h	16.2.12/ 751
2C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_33)	32	R/W	0009_0000h	16.2.12/ 751
2C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_34)	32	R/W	0009_0000h	16.2.12/ 751
2CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_35)	32	R/W	0009_0000h	16.2.12/ 751
2D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_36)	32	R/W	0009_0000h	16.2.12/ 751
2D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_37)	32	R/W	0009_0000h	16.2.12/ 751
2D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_38)	32	R/W	0009_0000h	16.2.12/ 751
2DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_39)	32	R/W	0009_0000h	16.2.12/ 751
2E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_40)	32	R/W	0009_0000h	16.2.12/ 751
2E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_41)	32	R/W	0009_0000h	16.2.12/ 751
2E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_42)	32	R/W	0009_0000h	16.2.12/ 751
2EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_43)	32	R/W	0009_0000h	16.2.12/ 751
2F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_44)	32	R/W	0009_0000h	16.2.12/ 751
2F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_45)	32	R/W	0009_0000h	16.2.12/ 751
2F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_46)	32	R/W	0009_0000h	16.2.12/ 751
2FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_47)	32	R/W	0009_0000h	16.2.12/ 751
300	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_48)	32	R/W	0009_0000h	16.2.12/ 751
304	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_49)	32	R/W	0009_0000h	16.2.12/ 751
308	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_50)	32	R/W	0009_0000h	16.2.12/ 751
30C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_51)	32	R/W	0009_0000h	16.2.12/ 751
310	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_52)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
314	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_53)	32	R/W	0009_0000h	16.2.12/ 751
318	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_54)	32	R/W	0009_0000h	16.2.12/ 751
31C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_55)	32	R/W	0009_0000h	16.2.12/ 751
320	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_56)	32	R/W	0009_0000h	16.2.12/ 751
324	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_57)	32	R/W	0009_0000h	16.2.12/ 751
328	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_58)	32	R/W	0009_0000h	16.2.12/ 751
32C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_59)	32	R/W	0009_0000h	16.2.12/ 751
330	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_60)	32	R/W	0009_0000h	16.2.12/ 751
334	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_61)	32	R/W	0009_0000h	16.2.12/ 751
338	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_62)	32	R/W	0009_0000h	16.2.12/ 751
33C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_63)	32	R/W	0009_0000h	16.2.12/ 751
340	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_64)	32	R/W	0009_0000h	16.2.12/ 751
344	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_65)	32	R/W	0009_0000h	16.2.12/ 751
348	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_66)	32	R/W	0009_0000h	16.2.12/ 751
34C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_67)	32	R/W	0009_0000h	16.2.12/ 751
350	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_68)	32	R/W	0009_0000h	16.2.12/ 751
354	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_69)	32	R/W	0009_0000h	16.2.12/ 751
358	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_70)	32	R/W	0009_0000h	16.2.12/ 751
35C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_71)	32	R/W	0009_0000h	16.2.12/ 751
360	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_72)	32	R/W	0009_0000h	16.2.12/ 751
364	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_73)	32	R/W	0009_0000h	16.2.12/ 751
368	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_74)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
36C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_75)	32	R/W	0009_0000h	16.2.12/ 751
370	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_76)	32	R/W	0009_0000h	16.2.12/ 751
374	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_77)	32	R/W	0009_0000h	16.2.12/ 751
378	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_78)	32	R/W	0009_0000h	16.2.12/ 751
37C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_79)	32	R/W	0009_0000h	16.2.12/ 751
380	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_80)	32	R/W	0009_0000h	16.2.12/ 751
384	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_81)	32	R/W	0009_0000h	16.2.12/ 751
388	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_82)	32	R/W	0009_0000h	16.2.12/ 751
38C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_83)	32	R/W	0009_0000h	16.2.12/ 751
390	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_84)	32	R/W	0009_0000h	16.2.12/ 751
394	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_85)	32	R/W	0009_0000h	16.2.12/ 751
398	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_86)	32	R/W	0009_0000h	16.2.12/ 751
39C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_87)	32	R/W	0009_0000h	16.2.12/ 751
3A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_88)	32	R/W	0009_0000h	16.2.12/ 751
3A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_89)	32	R/W	0009_0000h	16.2.12/ 751
3A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_90)	32	R/W	0009_0000h	16.2.12/ 751
3AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_91)	32	R/W	0009_0000h	16.2.12/ 751
3B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_92)	32	R/W	0009_0000h	16.2.12/ 751
3B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_93)	32	R/W	0009_0000h	16.2.12/ 751
3B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_94)	32	R/W	0009_0000h	16.2.12/ 751
3BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_95)	32	R/W	0009_0000h	16.2.12/ 751
3C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_96)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_97)	32	R/W	0009_0000h	16.2.12/ 751
3C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_98)	32	R/W	0009_0000h	16.2.12/ 751
3CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_99)	32	R/W	0009_0000h	16.2.12/ 751
3D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_100)	32	R/W	0009_0000h	16.2.12/ 751
3D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_101)	32	R/W	0009_0000h	16.2.12/ 751
3D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_102)	32	R/W	0009_0000h	16.2.12/ 751
3DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_103)	32	R/W	0009_0000h	16.2.12/ 751
3E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_104)	32	R/W	0009_0000h	16.2.12/ 751
3E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_105)	32	R/W	0009_0000h	16.2.12/ 751
3E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_106)	32	R/W	0009_0000h	16.2.12/ 751
3EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_107)	32	R/W	0009_0000h	16.2.12/ 751
3F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_108)	32	R/W	0009_0000h	16.2.12/ 751
3F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_109)	32	R/W	0009_0000h	16.2.12/ 751
3F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_110)	32	R/W	0009_0000h	16.2.12/ 751
3FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_111)	32	R/W	0009_0000h	16.2.12/ 751
400	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_112)	32	R/W	0009_0000h	16.2.12/ 751
404	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_113)	32	R/W	0009_0000h	16.2.12/ 751
408	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_114)	32	R/W	0009_0000h	16.2.12/ 751
40C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_115)	32	R/W	0009_0000h	16.2.12/ 751
410	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_116)	32	R/W	0009_0000h	16.2.12/ 751
414	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_117)	32	R/W	0009_0000h	16.2.12/ 751
418	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_118)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
41C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_119)	32	R/W	0009_0000h	16.2.12/ 751
420	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_120)	32	R/W	0009_0000h	16.2.12/ 751
424	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_121)	32	R/W	0009_0000h	16.2.12/ 751
428	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_122)	32	R/W	0009_0000h	16.2.12/ 751
42C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_123)	32	R/W	0009_0000h	16.2.12/ 751
430	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_124)	32	R/W	0009_0000h	16.2.12/ 751
434	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_125)	32	R/W	0009_0000h	16.2.12/ 751
438	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_126)	32	R/W	0009_0000h	16.2.12/ 751
43C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_127)	32	R/W	0009_0000h	16.2.12/ 751
440	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_128)	32	R/W	0009_0000h	16.2.12/ 751
444	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_129)	32	R/W	0009_0000h	16.2.12/ 751
448	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_130)	32	R/W	0009_0000h	16.2.12/ 751
44C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_131)	32	R/W	0009_0000h	16.2.12/ 751
450	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_132)	32	R/W	0009_0000h	16.2.12/ 751
454	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_133)	32	R/W	0009_0000h	16.2.12/ 751
458	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_134)	32	R/W	0009_0000h	16.2.12/ 751
45C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_135)	32	R/W	0009_0000h	16.2.12/ 751
460	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_136)	32	R/W	0009_0000h	16.2.12/ 751
464	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_137)	32	R/W	0009_0000h	16.2.12/ 751
468	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_138)	32	R/W	0009_0000h	16.2.12/ 751
46C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_139)	32	R/W	0009_0000h	16.2.12/ 751
470	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_140)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
474	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_141)	32	R/W	0009_0000h	16.2.12/ 751
478	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_142)	32	R/W	0009_0000h	16.2.12/ 751
47C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_143)	32	R/W	0009_0000h	16.2.12/ 751
480	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_144)	32	R/W	0009_0000h	16.2.12/ 751
484	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_145)	32	R/W	0009_0000h	16.2.12/ 751
488	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_146)	32	R/W	0009_0000h	16.2.12/ 751
48C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_147)	32	R/W	0009_0000h	16.2.12/ 751
490	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_148)	32	R/W	0009_0000h	16.2.12/ 751
494	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_149)	32	R/W	0009_0000h	16.2.12/ 751
498	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_150)	32	R/W	0009_0000h	16.2.12/ 751
49C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_151)	32	R/W	0009_0000h	16.2.12/ 751
4A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_152)	32	R/W	0009_0000h	16.2.12/ 751
4A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_153)	32	R/W	0009_0000h	16.2.12/ 751
4A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_154)	32	R/W	0009_0000h	16.2.12/ 751
4AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_155)	32	R/W	0009_0000h	16.2.12/ 751
4B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_156)	32	R/W	0009_0000h	16.2.12/ 751
4B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_157)	32	R/W	0009_0000h	16.2.12/ 751
4B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_158)	32	R/W	0009_0000h	16.2.12/ 751
4BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_159)	32	R/W	0009_0000h	16.2.12/ 751
4C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_160)	32	R/W	0009_0000h	16.2.12/ 751
4C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_161)	32	R/W	0009_0000h	16.2.12/ 751
4C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_162)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_163)	32	R/W	0009_0000h	16.2.12/ 751
4D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_164)	32	R/W	0009_0000h	16.2.12/ 751
4D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_165)	32	R/W	0009_0000h	16.2.12/ 751
4D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_166)	32	R/W	0009_0000h	16.2.12/ 751
4DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_167)	32	R/W	0009_0000h	16.2.12/ 751
4E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_168)	32	R/W	0009_0000h	16.2.12/ 751
4E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_169)	32	R/W	0009_0000h	16.2.12/ 751
4E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_170)	32	R/W	0009_0000h	16.2.12/ 751
4EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_171)	32	R/W	0009_0000h	16.2.12/ 751
4F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_172)	32	R/W	0009_0000h	16.2.12/ 751
4F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_173)	32	R/W	0009_0000h	16.2.12/ 751
4F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_174)	32	R/W	0009_0000h	16.2.12/ 751
4FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_175)	32	R/W	0009_0000h	16.2.12/ 751
500	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_176)	32	R/W	0009_0000h	16.2.12/ 751
504	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_177)	32	R/W	0009_0000h	16.2.12/ 751
508	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_178)	32	R/W	0009_0000h	16.2.12/ 751
50C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_179)	32	R/W	0009_0000h	16.2.12/ 751
510	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_180)	32	R/W	0009_0000h	16.2.12/ 751
514	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_181)	32	R/W	0009_0000h	16.2.12/ 751
518	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_182)	32	R/W	0009_0000h	16.2.12/ 751
51C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_183)	32	R/W	0009_0000h	16.2.12/ 751
520	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_184)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
524	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_185)	32	R/W	0009_0000h	16.2.12/ 751
528	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_186)	32	R/W	0009_0000h	16.2.12/ 751
52C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_187)	32	R/W	0009_0000h	16.2.12/ 751
530	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_188)	32	R/W	0009_0000h	16.2.12/ 751
534	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_189)	32	R/W	0009_0000h	16.2.12/ 751
538	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_190)	32	R/W	0009_0000h	16.2.12/ 751
53C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_191)	32	R/W	0009_0000h	16.2.12/ 751
540	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_192)	32	R/W	0009_0000h	16.2.12/ 751
544	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_193)	32	R/W	0009_0000h	16.2.12/ 751
548	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_194)	32	R/W	0009_0000h	16.2.12/ 751
54C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_195)	32	R/W	0009_0000h	16.2.12/ 751
550	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_196)	32	R/W	0009_0000h	16.2.12/ 751
554	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_197)	32	R/W	0009_0000h	16.2.12/ 751
558	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_198)	32	R/W	0009_0000h	16.2.12/ 751
55C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_199)	32	R/W	0009_0000h	16.2.12/ 751
560	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_200)	32	R/W	0009_0000h	16.2.12/ 751
564	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_201)	32	R/W	0009_0000h	16.2.12/ 751
568	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_202)	32	R/W	0009_0000h	16.2.12/ 751
56C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_203)	32	R/W	0009_0000h	16.2.12/ 751
570	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_204)	32	R/W	0009_0000h	16.2.12/ 751
574	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_205)	32	R/W	0009_0000h	16.2.12/ 751
578	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_206)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
57C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_207)	32	R/W	0009_0000h	16.2.12/ 751
580	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_208)	32	R/W	0009_0000h	16.2.12/ 751
584	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_209)	32	R/W	0009_0000h	16.2.12/ 751
588	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_210)	32	R/W	0009_0000h	16.2.12/ 751
58C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_211)	32	R/W	0009_0000h	16.2.12/ 751
590	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_212)	32	R/W	0009_0000h	16.2.12/ 751
594	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_213)	32	R/W	0009_0000h	16.2.12/ 751
598	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_214)	32	R/W	0009_0000h	16.2.12/ 751
59C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_215)	32	R/W	0009_0000h	16.2.12/ 751
5A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_216)	32	R/W	0009_0000h	16.2.12/ 751
5A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_217)	32	R/W	0009_0000h	16.2.12/ 751
5A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_218)	32	R/W	0009_0000h	16.2.12/ 751
5AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_219)	32	R/W	0009_0000h	16.2.12/ 751
5B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_220)	32	R/W	0009_0000h	16.2.12/ 751
5B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_221)	32	R/W	0009_0000h	16.2.12/ 751
5B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_222)	32	R/W	0009_0000h	16.2.12/ 751
5BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_223)	32	R/W	0009_0000h	16.2.12/ 751
5C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_224)	32	R/W	0009_0000h	16.2.12/ 751
5C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_225)	32	R/W	0009_0000h	16.2.12/ 751
5C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_226)	32	R/W	0009_0000h	16.2.12/ 751
5CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_227)	32	R/W	0009_0000h	16.2.12/ 751
5D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_228)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_229)	32	R/W	0009_0000h	16.2.12/ 751
5D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_230)	32	R/W	0009_0000h	16.2.12/ 751
5DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_231)	32	R/W	0009_0000h	16.2.12/ 751
5E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_232)	32	R/W	0009_0000h	16.2.12/ 751
5E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_233)	32	R/W	0009_0000h	16.2.12/ 751
5E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_234)	32	R/W	0009_0000h	16.2.12/ 751
5EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_235)	32	R/W	0009_0000h	16.2.12/ 751
5F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_236)	32	R/W	0009_0000h	16.2.12/ 751
5F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_237)	32	R/W	0009_0000h	16.2.12/ 751
5F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_238)	32	R/W	0009_0000h	16.2.12/ 751
5FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_239)	32	R/W	0009_0000h	16.2.12/ 751
600	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_240)	32	R/W	0009_0000h	16.2.12/ 751
604	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_241)	32	R/W	0009_0000h	16.2.12/ 751
608	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_242)	32	R/W	0009_0000h	16.2.12/ 751
60C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_243)	32	R/W	0009_0000h	16.2.12/ 751
610	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_244)	32	R/W	0009_0000h	16.2.12/ 751
614	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_245)	32	R/W	0009_0000h	16.2.12/ 751
618	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_246)	32	R/W	0009_0000h	16.2.12/ 751
61C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_247)	32	R/W	0009_0000h	16.2.12/ 751
620	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_248)	32	R/W	0009_0000h	16.2.12/ 751
624	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_249)	32	R/W	0009_0000h	16.2.12/ 751
628	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_250)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
62C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_251)	32	R/W	0009_0000h	16.2.12/ 751
630	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_252)	32	R/W	0009_0000h	16.2.12/ 751
634	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_253)	32	R/W	0009_0000h	16.2.12/ 751
638	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_254)	32	R/W	0009_0000h	16.2.12/ 751
63C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_255)	32	R/W	0009_0000h	16.2.12/ 751
640	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_256)	32	R/W	0009_0000h	16.2.12/ 751
644	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_257)	32	R/W	0009_0000h	16.2.12/ 751
648	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_258)	32	R/W	0009_0000h	16.2.12/ 751
64C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_259)	32	R/W	0009_0000h	16.2.12/ 751
650	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_260)	32	R/W	0009_0000h	16.2.12/ 751
654	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_261)	32	R/W	0009_0000h	16.2.12/ 751
658	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_262)	32	R/W	0009_0000h	16.2.12/ 751
65C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_263)	32	R/W	0009_0000h	16.2.12/ 751
660	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_264)	32	R/W	0009_0000h	16.2.12/ 751
664	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_265)	32	R/W	0009_0000h	16.2.12/ 751
668	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_266)	32	R/W	0009_0000h	16.2.12/ 751
66C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_267)	32	R/W	0009_0000h	16.2.12/ 751
670	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_268)	32	R/W	0009_0000h	16.2.12/ 751
674	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_269)	32	R/W	0009_0000h	16.2.12/ 751
678	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_270)	32	R/W	0009_0000h	16.2.12/ 751
67C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_271)	32	R/W	0009_0000h	16.2.12/ 751
680	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_272)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
684	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_273)	32	R/W	0009_0000h	16.2.12/ 751
688	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_274)	32	R/W	0009_0000h	16.2.12/ 751
68C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_275)	32	R/W	0009_0000h	16.2.12/ 751
690	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_276)	32	R/W	0009_0000h	16.2.12/ 751
694	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_277)	32	R/W	0009_0000h	16.2.12/ 751
698	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_278)	32	R/W	0009_0000h	16.2.12/ 751
69C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_279)	32	R/W	0009_0000h	16.2.12/ 751
6A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_280)	32	R/W	0009_0000h	16.2.12/ 751
6A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_281)	32	R/W	0009_0000h	16.2.12/ 751
6A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_282)	32	R/W	0009_0000h	16.2.12/ 751
6AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_283)	32	R/W	0009_0000h	16.2.12/ 751
6B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_284)	32	R/W	0009_0000h	16.2.12/ 751
6B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_285)	32	R/W	0009_0000h	16.2.12/ 751
6B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_286)	32	R/W	0009_0000h	16.2.12/ 751
6BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_287)	32	R/W	0009_0000h	16.2.12/ 751
6C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_288)	32	R/W	0009_0000h	16.2.12/ 751
6C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_289)	32	R/W	0009_0000h	16.2.12/ 751
6C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_290)	32	R/W	0009_0000h	16.2.12/ 751
6CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_291)	32	R/W	0009_0000h	16.2.12/ 751
6D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_292)	32	R/W	0009_0000h	16.2.12/ 751
6D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_293)	32	R/W	0009_0000h	16.2.12/ 751
6D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_294)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_295)	32	R/W	0009_0000h	16.2.12/ 751
6E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_296)	32	R/W	0009_0000h	16.2.12/ 751
6E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_297)	32	R/W	0009_0000h	16.2.12/ 751
6E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_298)	32	R/W	0009_0000h	16.2.12/ 751
6EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_299)	32	R/W	0009_0000h	16.2.12/ 751
6F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_300)	32	R/W	0009_0000h	16.2.12/ 751
6F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_301)	32	R/W	0009_0000h	16.2.12/ 751
6F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_302)	32	R/W	0009_0000h	16.2.12/ 751
6FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_303)	32	R/W	0009_0000h	16.2.12/ 751
700	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_304)	32	R/W	0009_0000h	16.2.12/ 751
704	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_305)	32	R/W	0009_0000h	16.2.12/ 751
708	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_306)	32	R/W	0009_0000h	16.2.12/ 751
70C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_307)	32	R/W	0009_0000h	16.2.12/ 751
710	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_308)	32	R/W	0009_0000h	16.2.12/ 751
714	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_309)	32	R/W	0009_0000h	16.2.12/ 751
718	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_310)	32	R/W	0009_0000h	16.2.12/ 751
71C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_311)	32	R/W	0009_0000h	16.2.12/ 751
720	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_312)	32	R/W	0009_0000h	16.2.12/ 751
724	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_313)	32	R/W	0009_0000h	16.2.12/ 751
728	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_314)	32	R/W	0009_0000h	16.2.12/ 751
72C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_315)	32	R/W	0009_0000h	16.2.12/ 751
730	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_316)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
734	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_317)	32	R/W	0009_0000h	16.2.12/ 751
738	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_318)	32	R/W	0009_0000h	16.2.12/ 751
73C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_319)	32	R/W	0009_0000h	16.2.12/ 751
740	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_320)	32	R/W	0009_0000h	16.2.12/ 751
744	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_321)	32	R/W	0009_0000h	16.2.12/ 751
748	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_322)	32	R/W	0009_0000h	16.2.12/ 751
74C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_323)	32	R/W	0009_0000h	16.2.12/ 751
750	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_324)	32	R/W	0009_0000h	16.2.12/ 751
754	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_325)	32	R/W	0009_0000h	16.2.12/ 751
758	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_326)	32	R/W	0009_0000h	16.2.12/ 751
75C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_327)	32	R/W	0009_0000h	16.2.12/ 751
760	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_328)	32	R/W	0009_0000h	16.2.12/ 751
764	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_329)	32	R/W	0009_0000h	16.2.12/ 751
768	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_330)	32	R/W	0009_0000h	16.2.12/ 751
76C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_331)	32	R/W	0009_0000h	16.2.12/ 751
770	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_332)	32	R/W	0009_0000h	16.2.12/ 751
774	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_333)	32	R/W	0009_0000h	16.2.12/ 751
778	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_334)	32	R/W	0009_0000h	16.2.12/ 751
77C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_335)	32	R/W	0009_0000h	16.2.12/ 751
780	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_336)	32	R/W	0009_0000h	16.2.12/ 751
784	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_337)	32	R/W	0009_0000h	16.2.12/ 751
788	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_338)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
78C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_339)	32	R/W	0009_0000h	16.2.12/ 751
790	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_340)	32	R/W	0009_0000h	16.2.12/ 751
794	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_341)	32	R/W	0009_0000h	16.2.12/ 751
798	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_342)	32	R/W	0009_0000h	16.2.12/ 751
79C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_343)	32	R/W	0009_0000h	16.2.12/ 751
7A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_344)	32	R/W	0009_0000h	16.2.12/ 751
7A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_345)	32	R/W	0009_0000h	16.2.12/ 751
7A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_346)	32	R/W	0009_0000h	16.2.12/ 751
7AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_347)	32	R/W	0009_0000h	16.2.12/ 751
7B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_348)	32	R/W	0009_0000h	16.2.12/ 751
7B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_349)	32	R/W	0009_0000h	16.2.12/ 751
7B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_350)	32	R/W	0009_0000h	16.2.12/ 751
7BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_351)	32	R/W	0009_0000h	16.2.12/ 751
7C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_352)	32	R/W	0009_0000h	16.2.12/ 751
7C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_353)	32	R/W	0009_0000h	16.2.12/ 751
7C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_354)	32	R/W	0009_0000h	16.2.12/ 751
7CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_355)	32	R/W	0009_0000h	16.2.12/ 751
7D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_356)	32	R/W	0009_0000h	16.2.12/ 751
7D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_357)	32	R/W	0009_0000h	16.2.12/ 751
7D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_358)	32	R/W	0009_0000h	16.2.12/ 751
7DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_359)	32	R/W	0009_0000h	16.2.12/ 751
7E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_360)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_361)	32	R/W	0009_0000h	16.2.12/ 751
7E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_362)	32	R/W	0009_0000h	16.2.12/ 751
7EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_363)	32	R/W	0009_0000h	16.2.12/ 751
7F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_364)	32	R/W	0009_0000h	16.2.12/ 751
7F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_365)	32	R/W	0009_0000h	16.2.12/ 751
7F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_366)	32	R/W	0009_0000h	16.2.12/ 751
7FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_367)	32	R/W	0009_0000h	16.2.12/ 751
800	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_368)	32	R/W	0009_0000h	16.2.12/ 751
804	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_369)	32	R/W	0009_0000h	16.2.12/ 751
808	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_370)	32	R/W	0009_0000h	16.2.12/ 751
80C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_371)	32	R/W	0009_0000h	16.2.12/ 751
810	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_372)	32	R/W	0009_0000h	16.2.12/ 751
814	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_373)	32	R/W	0009_0000h	16.2.12/ 751
818	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_374)	32	R/W	0009_0000h	16.2.12/ 751
81C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_375)	32	R/W	0009_0000h	16.2.12/ 751
820	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_376)	32	R/W	0009_0000h	16.2.12/ 751
824	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_377)	32	R/W	0009_0000h	16.2.12/ 751
828	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_378)	32	R/W	0009_0000h	16.2.12/ 751
82C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_379)	32	R/W	0009_0000h	16.2.12/ 751
830	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_380)	32	R/W	0009_0000h	16.2.12/ 751
834	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_381)	32	R/W	0009_0000h	16.2.12/ 751
838	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_382)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
83C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_383)	32	R/W	0009_0000h	16.2.12/ 751
840	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_384)	32	R/W	0009_0000h	16.2.12/ 751
844	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_385)	32	R/W	0009_0000h	16.2.12/ 751
848	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_386)	32	R/W	0009_0000h	16.2.12/ 751
84C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_387)	32	R/W	0009_0000h	16.2.12/ 751
850	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_388)	32	R/W	0009_0000h	16.2.12/ 751
854	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_389)	32	R/W	0009_0000h	16.2.12/ 751
858	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_390)	32	R/W	0009_0000h	16.2.12/ 751
85C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_391)	32	R/W	0009_0000h	16.2.12/ 751
860	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_392)	32	R/W	0009_0000h	16.2.12/ 751
864	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_393)	32	R/W	0009_0000h	16.2.12/ 751
868	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_394)	32	R/W	0009_0000h	16.2.12/ 751
86C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_395)	32	R/W	0009_0000h	16.2.12/ 751
870	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_396)	32	R/W	0009_0000h	16.2.12/ 751
874	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_397)	32	R/W	0009_0000h	16.2.12/ 751
878	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_398)	32	R/W	0009_0000h	16.2.12/ 751
87C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_399)	32	R/W	0009_0000h	16.2.12/ 751
880	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_400)	32	R/W	0009_0000h	16.2.12/ 751
884	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_401)	32	R/W	0009_0000h	16.2.12/ 751
888	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_402)	32	R/W	0009_0000h	16.2.12/ 751
88C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_403)	32	R/W	0009_0000h	16.2.12/ 751
890	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_404)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
894	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_405)	32	R/W	0009_0000h	16.2.12/ 751
898	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_406)	32	R/W	0009_0000h	16.2.12/ 751
89C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_407)	32	R/W	0009_0000h	16.2.12/ 751
8A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_408)	32	R/W	0009_0000h	16.2.12/ 751
8A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_409)	32	R/W	0009_0000h	16.2.12/ 751
8A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_410)	32	R/W	0009_0000h	16.2.12/ 751
8AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_411)	32	R/W	0009_0000h	16.2.12/ 751
8B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_412)	32	R/W	0009_0000h	16.2.12/ 751
8B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_413)	32	R/W	0009_0000h	16.2.12/ 751
8B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_414)	32	R/W	0009_0000h	16.2.12/ 751
8BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_415)	32	R/W	0009_0000h	16.2.12/ 751
8C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_416)	32	R/W	0009_0000h	16.2.12/ 751
8C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_417)	32	R/W	0009_0000h	16.2.12/ 751
8C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_418)	32	R/W	0009_0000h	16.2.12/ 751
8CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_419)	32	R/W	0009_0000h	16.2.12/ 751
8D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_420)	32	R/W	0009_0000h	16.2.12/ 751
8D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_421)	32	R/W	0009_0000h	16.2.12/ 751
8D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_422)	32	R/W	0009_0000h	16.2.12/ 751
8DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_423)	32	R/W	0009_0000h	16.2.12/ 751
8E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_424)	32	R/W	0009_0000h	16.2.12/ 751
8E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_425)	32	R/W	0009_0000h	16.2.12/ 751
8E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_426)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_427)	32	R/W	0009_0000h	16.2.12/ 751
8F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_428)	32	R/W	0009_0000h	16.2.12/ 751
8F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_429)	32	R/W	0009_0000h	16.2.12/ 751
8F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_430)	32	R/W	0009_0000h	16.2.12/ 751
8FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_431)	32	R/W	0009_0000h	16.2.12/ 751
900	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_432)	32	R/W	0009_0000h	16.2.12/ 751
904	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_433)	32	R/W	0009_0000h	16.2.12/ 751
908	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_434)	32	R/W	0009_0000h	16.2.12/ 751
90C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_435)	32	R/W	0009_0000h	16.2.12/ 751
910	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_436)	32	R/W	0009_0000h	16.2.12/ 751
914	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_437)	32	R/W	0009_0000h	16.2.12/ 751
918	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_438)	32	R/W	0009_0000h	16.2.12/ 751
91C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_439)	32	R/W	0009_0000h	16.2.12/ 751
920	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_440)	32	R/W	0009_0000h	16.2.12/ 751
924	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_441)	32	R/W	0009_0000h	16.2.12/ 751
928	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_442)	32	R/W	0009_0000h	16.2.12/ 751
92C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_443)	32	R/W	0009_0000h	16.2.12/ 751
930	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_444)	32	R/W	0009_0000h	16.2.12/ 751
934	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_445)	32	R/W	0009_0000h	16.2.12/ 751
938	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_446)	32	R/W	0009_0000h	16.2.12/ 751
93C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_447)	32	R/W	0009_0000h	16.2.12/ 751
940	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_448)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
944	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_449)	32	R/W	0009_0000h	16.2.12/ 751
948	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_450)	32	R/W	0009_0000h	16.2.12/ 751
94C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_451)	32	R/W	0009_0000h	16.2.12/ 751
950	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_452)	32	R/W	0009_0000h	16.2.12/ 751
954	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_453)	32	R/W	0009_0000h	16.2.12/ 751
958	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_454)	32	R/W	0009_0000h	16.2.12/ 751
95C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_455)	32	R/W	0009_0000h	16.2.12/ 751
960	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_456)	32	R/W	0009_0000h	16.2.12/ 751
964	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_457)	32	R/W	0009_0000h	16.2.12/ 751
968	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_458)	32	R/W	0009_0000h	16.2.12/ 751
96C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_459)	32	R/W	0009_0000h	16.2.12/ 751
970	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_460)	32	R/W	0009_0000h	16.2.12/ 751
974	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_461)	32	R/W	0009_0000h	16.2.12/ 751
978	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_462)	32	R/W	0009_0000h	16.2.12/ 751
97C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_463)	32	R/W	0009_0000h	16.2.12/ 751
980	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_464)	32	R/W	0009_0000h	16.2.12/ 751
984	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_465)	32	R/W	0009_0000h	16.2.12/ 751
988	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_466)	32	R/W	0009_0000h	16.2.12/ 751
98C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_467)	32	R/W	0009_0000h	16.2.12/ 751
990	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_468)	32	R/W	0009_0000h	16.2.12/ 751
994	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_469)	32	R/W	0009_0000h	16.2.12/ 751
998	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_470)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
99C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_471)	32	R/W	0009_0000h	16.2.12/ 751
9A0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_472)	32	R/W	0009_0000h	16.2.12/ 751
9A4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_473)	32	R/W	0009_0000h	16.2.12/ 751
9A8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_474)	32	R/W	0009_0000h	16.2.12/ 751
9AC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_475)	32	R/W	0009_0000h	16.2.12/ 751
9B0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_476)	32	R/W	0009_0000h	16.2.12/ 751
9B4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_477)	32	R/W	0009_0000h	16.2.12/ 751
9B8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_478)	32	R/W	0009_0000h	16.2.12/ 751
9BC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_479)	32	R/W	0009_0000h	16.2.12/ 751
9C0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_480)	32	R/W	0009_0000h	16.2.12/ 751
9C4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_481)	32	R/W	0009_0000h	16.2.12/ 751
9C8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_482)	32	R/W	0009_0000h	16.2.12/ 751
9CC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_483)	32	R/W	0009_0000h	16.2.12/ 751
9D0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_484)	32	R/W	0009_0000h	16.2.12/ 751
9D4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_485)	32	R/W	0009_0000h	16.2.12/ 751
9D8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_486)	32	R/W	0009_0000h	16.2.12/ 751
9DC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_487)	32	R/W	0009_0000h	16.2.12/ 751
9E0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_488)	32	R/W	0009_0000h	16.2.12/ 751
9E4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_489)	32	R/W	0009_0000h	16.2.12/ 751
9E8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_490)	32	R/W	0009_0000h	16.2.12/ 751
9EC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_491)	32	R/W	0009_0000h	16.2.12/ 751
9F0	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_492)	32	R/W	0009_0000h	16.2.12/ 751

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9F4	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_493)	32	R/W	0009_0000h	16.2.12/ 751
9F8	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_494)	32	R/W	0009_0000h	16.2.12/ 751
9FC	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_495)	32	R/W	0009_0000h	16.2.12/ 751
A00	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_496)	32	R/W	0009_0000h	16.2.12/ 751
A04	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_497)	32	R/W	0009_0000h	16.2.12/ 751
A08	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_498)	32	R/W	0009_0000h	16.2.12/ 751
A0C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_499)	32	R/W	0009_0000h	16.2.12/ 751
A10	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_500)	32	R/W	0009_0000h	16.2.12/ 751
A14	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_501)	32	R/W	0009_0000h	16.2.12/ 751
A18	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_502)	32	R/W	0009_0000h	16.2.12/ 751
A1C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_503)	32	R/W	0009_0000h	16.2.12/ 751
A20	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_504)	32	R/W	0009_0000h	16.2.12/ 751
A24	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_505)	32	R/W	0009_0000h	16.2.12/ 751
A28	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_506)	32	R/W	0009_0000h	16.2.12/ 751
A2C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_507)	32	R/W	0009_0000h	16.2.12/ 751
A30	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_508)	32	R/W	0009_0000h	16.2.12/ 751
A34	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_509)	32	R/W	0009_0000h	16.2.12/ 751
A38	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_510)	32	R/W	0009_0000h	16.2.12/ 751
A3C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_511)	32	R/W	0009_0000h	16.2.12/ 751
A40	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_512)	32	R/W	0000_0000h	16.2.13/ 754
A44	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_513)	32	R/W	0000_0000h	16.2.13/ 754
A48	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_514)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_515)	32	R/W	0000_0000h	16.2.13/ 754
A50	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_516)	32	R/W	0000_0000h	16.2.13/ 754
A54	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_517)	32	R/W	0000_0000h	16.2.13/ 754
A58	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_518)	32	R/W	0000_0000h	16.2.13/ 754
A5C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_519)	32	R/W	0000_0000h	16.2.13/ 754
A60	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_520)	32	R/W	0000_0000h	16.2.13/ 754
A64	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_521)	32	R/W	0000_0000h	16.2.13/ 754
A68	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_522)	32	R/W	0000_0000h	16.2.13/ 754
A6C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_523)	32	R/W	0000_0000h	16.2.13/ 754
A70	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_524)	32	R/W	0000_0000h	16.2.13/ 754
A74	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_525)	32	R/W	0000_0000h	16.2.13/ 754
A78	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_526)	32	R/W	0000_0000h	16.2.13/ 754
A7C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_527)	32	R/W	0000_0000h	16.2.13/ 754
A80	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_528)	32	R/W	0000_0000h	16.2.13/ 754
A84	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_529)	32	R/W	0000_0000h	16.2.13/ 754
A88	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_530)	32	R/W	0000_0000h	16.2.13/ 754
A8C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_531)	32	R/W	0000_0000h	16.2.13/ 754
A90	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_532)	32	R/W	0000_0000h	16.2.13/ 754
A94	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_533)	32	R/W	0000_0000h	16.2.13/ 754
A98	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_534)	32	R/W	0000_0000h	16.2.13/ 754
A9C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_535)	32	R/W	0000_0000h	16.2.13/ 754
AA0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_536)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AA4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_537)	32	R/W	0000_0000h	16.2.13/ 754
AA8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_538)	32	R/W	0000_0000h	16.2.13/ 754
AAC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_539)	32	R/W	0000_0000h	16.2.13/ 754
AB0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_540)	32	R/W	0000_0000h	16.2.13/ 754
AB4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_541)	32	R/W	0000_0000h	16.2.13/ 754
AB8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_542)	32	R/W	0000_0000h	16.2.13/ 754
ABC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_543)	32	R/W	0000_0000h	16.2.13/ 754
AC0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_544)	32	R/W	0000_0000h	16.2.13/ 754
AC4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_545)	32	R/W	0000_0000h	16.2.13/ 754
AC8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_546)	32	R/W	0000_0000h	16.2.13/ 754
ACC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_547)	32	R/W	0000_0000h	16.2.13/ 754
AD0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_548)	32	R/W	0000_0000h	16.2.13/ 754
AD4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_549)	32	R/W	0000_0000h	16.2.13/ 754
AD8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_550)	32	R/W	0000_0000h	16.2.13/ 754
ADC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_551)	32	R/W	0000_0000h	16.2.13/ 754
AE0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_552)	32	R/W	0000_0000h	16.2.13/ 754
AE4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_553)	32	R/W	0000_0000h	16.2.13/ 754
AE8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_554)	32	R/W	0000_0000h	16.2.13/ 754
AEC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_555)	32	R/W	0000_0000h	16.2.13/ 754
AF0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_556)	32	R/W	0000_0000h	16.2.13/ 754
AF4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_557)	32	R/W	0000_0000h	16.2.13/ 754
AF8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_558)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AFC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_559)	32	R/W	0000_0000h	16.2.13/ 754
B00	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_560)	32	R/W	0000_0000h	16.2.13/ 754
B04	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_561)	32	R/W	0000_0000h	16.2.13/ 754
B08	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_562)	32	R/W	0000_0000h	16.2.13/ 754
B0C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_563)	32	R/W	0000_0000h	16.2.13/ 754
B10	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_564)	32	R/W	0000_0000h	16.2.13/ 754
B14	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_565)	32	R/W	0000_0000h	16.2.13/ 754
B18	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_566)	32	R/W	0000_0000h	16.2.13/ 754
B1C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_567)	32	R/W	0000_0000h	16.2.13/ 754
B20	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_568)	32	R/W	0000_0000h	16.2.13/ 754
B24	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_569)	32	R/W	0000_0000h	16.2.13/ 754
B28	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_570)	32	R/W	0000_0000h	16.2.13/ 754
B2C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_571)	32	R/W	0000_0000h	16.2.13/ 754
B30	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_572)	32	R/W	0000_0000h	16.2.13/ 754
B34	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_573)	32	R/W	0000_0000h	16.2.13/ 754
B38	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_574)	32	R/W	0000_0000h	16.2.13/ 754
B3C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_575)	32	R/W	0000_0000h	16.2.13/ 754
B40	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_576)	32	R/W	0000_0000h	16.2.13/ 754
B44	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_577)	32	R/W	0000_0000h	16.2.13/ 754
B48	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_578)	32	R/W	0000_0000h	16.2.13/ 754
B4C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_579)	32	R/W	0000_0000h	16.2.13/ 754
B50	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_580)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B54	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_581)	32	R/W	0000_0000h	16.2.13/ 754
B58	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_582)	32	R/W	0000_0000h	16.2.13/ 754
B5C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_583)	32	R/W	0000_0000h	16.2.13/ 754
B60	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_584)	32	R/W	0000_0000h	16.2.13/ 754
B64	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_585)	32	R/W	0000_0000h	16.2.13/ 754
B68	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_586)	32	R/W	0000_0000h	16.2.13/ 754
B6C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_587)	32	R/W	0000_0000h	16.2.13/ 754
B70	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_588)	32	R/W	0000_0000h	16.2.13/ 754
B74	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_589)	32	R/W	0000_0000h	16.2.13/ 754
B78	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_590)	32	R/W	0000_0000h	16.2.13/ 754
B7C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_591)	32	R/W	0000_0000h	16.2.13/ 754
B80	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_592)	32	R/W	0000_0000h	16.2.13/ 754
B84	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_593)	32	R/W	0000_0000h	16.2.13/ 754
B88	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_594)	32	R/W	0000_0000h	16.2.13/ 754
B8C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_595)	32	R/W	0000_0000h	16.2.13/ 754
B90	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_596)	32	R/W	0000_0000h	16.2.13/ 754
B94	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_597)	32	R/W	0000_0000h	16.2.13/ 754
B98	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_598)	32	R/W	0000_0000h	16.2.13/ 754
B9C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_599)	32	R/W	0000_0000h	16.2.13/ 754
BA0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_600)	32	R/W	0000_0000h	16.2.13/ 754
BA4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_601)	32	R/W	0000_0000h	16.2.13/ 754
BA8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_602)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BAC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_603)	32	R/W	0000_0000h	16.2.13/ 754
BB0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_604)	32	R/W	0000_0000h	16.2.13/ 754
BB4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_605)	32	R/W	0000_0000h	16.2.13/ 754
BB8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_606)	32	R/W	0000_0000h	16.2.13/ 754
BBC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_607)	32	R/W	0000_0000h	16.2.13/ 754
BC0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_608)	32	R/W	0000_0000h	16.2.13/ 754
BC4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_609)	32	R/W	0000_0000h	16.2.13/ 754
BC8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_610)	32	R/W	0000_0000h	16.2.13/ 754
BCC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_611)	32	R/W	0000_0000h	16.2.13/ 754
BD0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_612)	32	R/W	0000_0000h	16.2.13/ 754
BD4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_613)	32	R/W	0000_0000h	16.2.13/ 754
BD8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_614)	32	R/W	0000_0000h	16.2.13/ 754
BDC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_615)	32	R/W	0000_0000h	16.2.13/ 754
BE0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_616)	32	R/W	0000_0000h	16.2.13/ 754
BE4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_617)	32	R/W	0000_0000h	16.2.13/ 754
BE8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_618)	32	R/W	0000_0000h	16.2.13/ 754
BEC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_619)	32	R/W	0000_0000h	16.2.13/ 754
BF0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_620)	32	R/W	0000_0000h	16.2.13/ 754
BF4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_621)	32	R/W	0000_0000h	16.2.13/ 754
BF8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_622)	32	R/W	0000_0000h	16.2.13/ 754
BFC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_623)	32	R/W	0000_0000h	16.2.13/ 754
C00	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_624)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C04	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_625)	32	R/W	0000_0000h	16.2.13/ 754
C08	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_626)	32	R/W	0000_0000h	16.2.13/ 754
C0C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_627)	32	R/W	0000_0000h	16.2.13/ 754
C10	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_628)	32	R/W	0000_0000h	16.2.13/ 754
C14	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_629)	32	R/W	0000_0000h	16.2.13/ 754
C18	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_630)	32	R/W	0000_0000h	16.2.13/ 754
C1C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_631)	32	R/W	0000_0000h	16.2.13/ 754
C20	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_632)	32	R/W	0000_0000h	16.2.13/ 754
C24	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_633)	32	R/W	0000_0000h	16.2.13/ 754
C28	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_634)	32	R/W	0000_0000h	16.2.13/ 754
C2C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_635)	32	R/W	0000_0000h	16.2.13/ 754
C30	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_636)	32	R/W	0000_0000h	16.2.13/ 754
C34	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_637)	32	R/W	0000_0000h	16.2.13/ 754
C38	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_638)	32	R/W	0000_0000h	16.2.13/ 754
C3C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_639)	32	R/W	0000_0000h	16.2.13/ 754
C40	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_640)	32	R/W	0000_0000h	16.2.13/ 754
C44	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_641)	32	R/W	0000_0000h	16.2.13/ 754
C48	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_642)	32	R/W	0000_0000h	16.2.13/ 754
C4C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_643)	32	R/W	0000_0000h	16.2.13/ 754
C50	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_644)	32	R/W	0000_0000h	16.2.13/ 754
C54	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_645)	32	R/W	0000_0000h	16.2.13/ 754
C58	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_646)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C5C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_647)	32	R/W	0000_0000h	16.2.13/ 754
C60	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_648)	32	R/W	0000_0000h	16.2.13/ 754
C64	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_649)	32	R/W	0000_0000h	16.2.13/ 754
C68	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_650)	32	R/W	0000_0000h	16.2.13/ 754
C6C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_651)	32	R/W	0000_0000h	16.2.13/ 754
C70	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_652)	32	R/W	0000_0000h	16.2.13/ 754
C74	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_653)	32	R/W	0000_0000h	16.2.13/ 754
C78	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_654)	32	R/W	0000_0000h	16.2.13/ 754
C7C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_655)	32	R/W	0000_0000h	16.2.13/ 754
C80	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_656)	32	R/W	0000_0000h	16.2.13/ 754
C84	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_657)	32	R/W	0000_0000h	16.2.13/ 754
C88	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_658)	32	R/W	0000_0000h	16.2.13/ 754
C8C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_659)	32	R/W	0000_0000h	16.2.13/ 754
C90	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_660)	32	R/W	0000_0000h	16.2.13/ 754
C94	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_661)	32	R/W	0000_0000h	16.2.13/ 754
C98	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_662)	32	R/W	0000_0000h	16.2.13/ 754
C9C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_663)	32	R/W	0000_0000h	16.2.13/ 754
CA0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_664)	32	R/W	0000_0000h	16.2.13/ 754
CA4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_665)	32	R/W	0000_0000h	16.2.13/ 754
CA8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_666)	32	R/W	0000_0000h	16.2.13/ 754
CAC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_667)	32	R/W	0000_0000h	16.2.13/ 754
CB0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_668)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CB4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_669)	32	R/W	0000_0000h	16.2.13/ 754
CB8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_670)	32	R/W	0000_0000h	16.2.13/ 754
CBC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_671)	32	R/W	0000_0000h	16.2.13/ 754
CC0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_672)	32	R/W	0000_0000h	16.2.13/ 754
CC4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_673)	32	R/W	0000_0000h	16.2.13/ 754
CC8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_674)	32	R/W	0000_0000h	16.2.13/ 754
CCC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_675)	32	R/W	0000_0000h	16.2.13/ 754
CD0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_676)	32	R/W	0000_0000h	16.2.13/ 754
CD4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_677)	32	R/W	0000_0000h	16.2.13/ 754
CD8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_678)	32	R/W	0000_0000h	16.2.13/ 754
CDC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_679)	32	R/W	0000_0000h	16.2.13/ 754
CE0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_680)	32	R/W	0000_0000h	16.2.13/ 754
CE4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_681)	32	R/W	0000_0000h	16.2.13/ 754
CE8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_682)	32	R/W	0000_0000h	16.2.13/ 754
CEC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_683)	32	R/W	0000_0000h	16.2.13/ 754
CF0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_684)	32	R/W	0000_0000h	16.2.13/ 754
CF4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_685)	32	R/W	0000_0000h	16.2.13/ 754
CF8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_686)	32	R/W	0000_0000h	16.2.13/ 754
CFC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_687)	32	R/W	0000_0000h	16.2.13/ 754
D00	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_688)	32	R/W	0000_0000h	16.2.13/ 754
D04	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_689)	32	R/W	0000_0000h	16.2.13/ 754
D08	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_690)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D0C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_691)	32	R/W	0000_0000h	16.2.13/ 754
D10	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_692)	32	R/W	0000_0000h	16.2.13/ 754
D14	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_693)	32	R/W	0000_0000h	16.2.13/ 754
D18	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_694)	32	R/W	0000_0000h	16.2.13/ 754
D1C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_695)	32	R/W	0000_0000h	16.2.13/ 754
D20	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_696)	32	R/W	0000_0000h	16.2.13/ 754
D24	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_697)	32	R/W	0000_0000h	16.2.13/ 754
D28	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_698)	32	R/W	0000_0000h	16.2.13/ 754
D2C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_699)	32	R/W	0000_0000h	16.2.13/ 754
D30	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_700)	32	R/W	0000_0000h	16.2.13/ 754
D34	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_701)	32	R/W	0000_0000h	16.2.13/ 754
D38	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_702)	32	R/W	0000_0000h	16.2.13/ 754
D3C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_703)	32	R/W	0000_0000h	16.2.13/ 754
D40	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_704)	32	R/W	0000_0000h	16.2.13/ 754
D44	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_705)	32	R/W	0000_0000h	16.2.13/ 754
D48	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_706)	32	R/W	0000_0000h	16.2.13/ 754
D4C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_707)	32	R/W	0000_0000h	16.2.13/ 754
D50	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_708)	32	R/W	0000_0000h	16.2.13/ 754
D54	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_709)	32	R/W	0000_0000h	16.2.13/ 754
D58	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_710)	32	R/W	0000_0000h	16.2.13/ 754
D5C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_711)	32	R/W	0000_0000h	16.2.13/ 754
D60	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_712)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D64	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_713)	32	R/W	0000_0000h	16.2.13/ 754
D68	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_714)	32	R/W	0000_0000h	16.2.13/ 754
D6C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_715)	32	R/W	0000_0000h	16.2.13/ 754
D70	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_716)	32	R/W	0000_0000h	16.2.13/ 754
D74	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_717)	32	R/W	0000_0000h	16.2.13/ 754
D78	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_718)	32	R/W	0000_0000h	16.2.13/ 754
D7C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_719)	32	R/W	0000_0000h	16.2.13/ 754
D80	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_720)	32	R/W	0000_0000h	16.2.13/ 754
D84	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_721)	32	R/W	0000_0000h	16.2.13/ 754
D88	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_722)	32	R/W	0000_0000h	16.2.13/ 754
D8C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_723)	32	R/W	0000_0000h	16.2.13/ 754
D90	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_724)	32	R/W	0000_0000h	16.2.13/ 754
D94	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_725)	32	R/W	0000_0000h	16.2.13/ 754
D98	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_726)	32	R/W	0000_0000h	16.2.13/ 754
D9C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_727)	32	R/W	0000_0000h	16.2.13/ 754
DA0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_728)	32	R/W	0000_0000h	16.2.13/ 754
DA4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_729)	32	R/W	0000_0000h	16.2.13/ 754
DA8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_730)	32	R/W	0000_0000h	16.2.13/ 754
DAC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_731)	32	R/W	0000_0000h	16.2.13/ 754
DB0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_732)	32	R/W	0000_0000h	16.2.13/ 754
DB4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_733)	32	R/W	0000_0000h	16.2.13/ 754
DB8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_734)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DBC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_735)	32	R/W	0000_0000h	16.2.13/ 754
DC0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_736)	32	R/W	0000_0000h	16.2.13/ 754
DC4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_737)	32	R/W	0000_0000h	16.2.13/ 754
DC8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_738)	32	R/W	0000_0000h	16.2.13/ 754
DCC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_739)	32	R/W	0000_0000h	16.2.13/ 754
DD0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_740)	32	R/W	0000_0000h	16.2.13/ 754
DD4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_741)	32	R/W	0000_0000h	16.2.13/ 754
DD8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_742)	32	R/W	0000_0000h	16.2.13/ 754
DDC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_743)	32	R/W	0000_0000h	16.2.13/ 754
DE0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_744)	32	R/W	0000_0000h	16.2.13/ 754
DE4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_745)	32	R/W	0000_0000h	16.2.13/ 754
DE8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_746)	32	R/W	0000_0000h	16.2.13/ 754
DEC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_747)	32	R/W	0000_0000h	16.2.13/ 754
DF0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_748)	32	R/W	0000_0000h	16.2.13/ 754
DF4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_749)	32	R/W	0000_0000h	16.2.13/ 754
DF8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_750)	32	R/W	0000_0000h	16.2.13/ 754
DFC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_751)	32	R/W	0000_0000h	16.2.13/ 754
E00	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_752)	32	R/W	0000_0000h	16.2.13/ 754
E04	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_753)	32	R/W	0000_0000h	16.2.13/ 754
E08	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_754)	32	R/W	0000_0000h	16.2.13/ 754
E0C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_755)	32	R/W	0000_0000h	16.2.13/ 754
E10	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_756)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E14	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_757)	32	R/W	0000_0000h	16.2.13/ 754
E18	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_758)	32	R/W	0000_0000h	16.2.13/ 754
E1C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_759)	32	R/W	0000_0000h	16.2.13/ 754
E20	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_760)	32	R/W	0000_0000h	16.2.13/ 754
E24	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_761)	32	R/W	0000_0000h	16.2.13/ 754
E28	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_762)	32	R/W	0000_0000h	16.2.13/ 754
E2C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_763)	32	R/W	0000_0000h	16.2.13/ 754
E30	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_764)	32	R/W	0000_0000h	16.2.13/ 754
E34	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_765)	32	R/W	0000_0000h	16.2.13/ 754
E38	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_766)	32	R/W	0000_0000h	16.2.13/ 754
E3C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_767)	32	R/W	0000_0000h	16.2.13/ 754
E40	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_768)	32	R/W	0000_0000h	16.2.13/ 754
E44	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_769)	32	R/W	0000_0000h	16.2.13/ 754
E48	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_770)	32	R/W	0000_0000h	16.2.13/ 754
E4C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_771)	32	R/W	0000_0000h	16.2.13/ 754
E50	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_772)	32	R/W	0000_0000h	16.2.13/ 754
E54	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_773)	32	R/W	0000_0000h	16.2.13/ 754
E58	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_774)	32	R/W	0000_0000h	16.2.13/ 754
E5C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_775)	32	R/W	0000_0000h	16.2.13/ 754
E60	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_776)	32	R/W	0000_0000h	16.2.13/ 754
E64	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_777)	32	R/W	0000_0000h	16.2.13/ 754
E68	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_778)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E6C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_779)	32	R/W	0000_0000h	16.2.13/ 754
E70	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_780)	32	R/W	0000_0000h	16.2.13/ 754
E74	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_781)	32	R/W	0000_0000h	16.2.13/ 754
E78	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_782)	32	R/W	0000_0000h	16.2.13/ 754
E7C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_783)	32	R/W	0000_0000h	16.2.13/ 754
E80	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_784)	32	R/W	0000_0000h	16.2.13/ 754
E84	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_785)	32	R/W	0000_0000h	16.2.13/ 754
E88	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_786)	32	R/W	0000_0000h	16.2.13/ 754
E8C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_787)	32	R/W	0000_0000h	16.2.13/ 754
E90	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_788)	32	R/W	0000_0000h	16.2.13/ 754
E94	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_789)	32	R/W	0000_0000h	16.2.13/ 754
E98	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_790)	32	R/W	0000_0000h	16.2.13/ 754
E9C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_791)	32	R/W	0000_0000h	16.2.13/ 754
EA0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_792)	32	R/W	0000_0000h	16.2.13/ 754
EA4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_793)	32	R/W	0000_0000h	16.2.13/ 754
EA8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_794)	32	R/W	0000_0000h	16.2.13/ 754
EAC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_795)	32	R/W	0000_0000h	16.2.13/ 754
EB0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_796)	32	R/W	0000_0000h	16.2.13/ 754
EB4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_797)	32	R/W	0000_0000h	16.2.13/ 754
EB8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_798)	32	R/W	0000_0000h	16.2.13/ 754
EBC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_799)	32	R/W	0000_0000h	16.2.13/ 754
EC0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_800)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EC4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_801)	32	R/W	0000_0000h	16.2.13/ 754
EC8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_802)	32	R/W	0000_0000h	16.2.13/ 754
ECC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_803)	32	R/W	0000_0000h	16.2.13/ 754
ED0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_804)	32	R/W	0000_0000h	16.2.13/ 754
ED4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_805)	32	R/W	0000_0000h	16.2.13/ 754
ED8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_806)	32	R/W	0000_0000h	16.2.13/ 754
EDC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_807)	32	R/W	0000_0000h	16.2.13/ 754
EE0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_808)	32	R/W	0000_0000h	16.2.13/ 754
EE4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_809)	32	R/W	0000_0000h	16.2.13/ 754
EE8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_810)	32	R/W	0000_0000h	16.2.13/ 754
EEC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_811)	32	R/W	0000_0000h	16.2.13/ 754
EF0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_812)	32	R/W	0000_0000h	16.2.13/ 754
EF4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_813)	32	R/W	0000_0000h	16.2.13/ 754
EF8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_814)	32	R/W	0000_0000h	16.2.13/ 754
EFC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_815)	32	R/W	0000_0000h	16.2.13/ 754
F00	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_816)	32	R/W	0000_0000h	16.2.13/ 754
F04	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_817)	32	R/W	0000_0000h	16.2.13/ 754
F08	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_818)	32	R/W	0000_0000h	16.2.13/ 754
F0C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_819)	32	R/W	0000_0000h	16.2.13/ 754
F10	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_820)	32	R/W	0000_0000h	16.2.13/ 754
F14	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_821)	32	R/W	0000_0000h	16.2.13/ 754
F18	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_822)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F1C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_823)	32	R/W	0000_0000h	16.2.13/ 754
F20	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_824)	32	R/W	0000_0000h	16.2.13/ 754
F24	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_825)	32	R/W	0000_0000h	16.2.13/ 754
F28	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_826)	32	R/W	0000_0000h	16.2.13/ 754
F2C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_827)	32	R/W	0000_0000h	16.2.13/ 754
F30	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_828)	32	R/W	0000_0000h	16.2.13/ 754
F34	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_829)	32	R/W	0000_0000h	16.2.13/ 754
F38	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_830)	32	R/W	0000_0000h	16.2.13/ 754
F3C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_831)	32	R/W	0000_0000h	16.2.13/ 754
F40	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_832)	32	R/W	0000_0000h	16.2.13/ 754
F44	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_833)	32	R/W	0000_0000h	16.2.13/ 754
F48	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_834)	32	R/W	0000_0000h	16.2.13/ 754
F4C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_835)	32	R/W	0000_0000h	16.2.13/ 754
F50	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_836)	32	R/W	0000_0000h	16.2.13/ 754
F54	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_837)	32	R/W	0000_0000h	16.2.13/ 754
F58	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_838)	32	R/W	0000_0000h	16.2.13/ 754
F5C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_839)	32	R/W	0000_0000h	16.2.13/ 754
F60	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_840)	32	R/W	0000_0000h	16.2.13/ 754
F64	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_841)	32	R/W	0000_0000h	16.2.13/ 754
F68	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_842)	32	R/W	0000_0000h	16.2.13/ 754
F6C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_843)	32	R/W	0000_0000h	16.2.13/ 754
F70	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_844)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F74	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_845)	32	R/W	0000_0000h	16.2.13/ 754
F78	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_846)	32	R/W	0000_0000h	16.2.13/ 754
F7C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_847)	32	R/W	0000_0000h	16.2.13/ 754
F80	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_848)	32	R/W	0000_0000h	16.2.13/ 754
F84	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_849)	32	R/W	0000_0000h	16.2.13/ 754
F88	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_850)	32	R/W	0000_0000h	16.2.13/ 754
F8C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_851)	32	R/W	0000_0000h	16.2.13/ 754
F90	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_852)	32	R/W	0000_0000h	16.2.13/ 754
F94	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_853)	32	R/W	0000_0000h	16.2.13/ 754
F98	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_854)	32	R/W	0000_0000h	16.2.13/ 754
F9C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_855)	32	R/W	0000_0000h	16.2.13/ 754
FA0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_856)	32	R/W	0000_0000h	16.2.13/ 754
FA4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_857)	32	R/W	0000_0000h	16.2.13/ 754
FA8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_858)	32	R/W	0000_0000h	16.2.13/ 754
FAC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_859)	32	R/W	0000_0000h	16.2.13/ 754
FB0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_860)	32	R/W	0000_0000h	16.2.13/ 754
FB4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_861)	32	R/W	0000_0000h	16.2.13/ 754
FB8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_862)	32	R/W	0000_0000h	16.2.13/ 754
FBC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_863)	32	R/W	0000_0000h	16.2.13/ 754
FC0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_864)	32	R/W	0000_0000h	16.2.13/ 754
FC4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_865)	32	R/W	0000_0000h	16.2.13/ 754
FC8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_866)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FCC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_867)	32	R/W	0000_0000h	16.2.13/ 754
FD0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_868)	32	R/W	0000_0000h	16.2.13/ 754
FD4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_869)	32	R/W	0000_0000h	16.2.13/ 754
FD8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_870)	32	R/W	0000_0000h	16.2.13/ 754
FDC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_871)	32	R/W	0000_0000h	16.2.13/ 754
FE0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_872)	32	R/W	0000_0000h	16.2.13/ 754
FE4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_873)	32	R/W	0000_0000h	16.2.13/ 754
FE8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_874)	32	R/W	0000_0000h	16.2.13/ 754
FEC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_875)	32	R/W	0000_0000h	16.2.13/ 754
FF0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_876)	32	R/W	0000_0000h	16.2.13/ 754
FF4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_877)	32	R/W	0000_0000h	16.2.13/ 754
FF8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_878)	32	R/W	0000_0000h	16.2.13/ 754
FFC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_879)	32	R/W	0000_0000h	16.2.13/ 754
1000	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_880)	32	R/W	0000_0000h	16.2.13/ 754
1004	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_881)	32	R/W	0000_0000h	16.2.13/ 754
1008	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_882)	32	R/W	0000_0000h	16.2.13/ 754
100C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_883)	32	R/W	0000_0000h	16.2.13/ 754
1010	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_884)	32	R/W	0000_0000h	16.2.13/ 754
1014	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_885)	32	R/W	0000_0000h	16.2.13/ 754
1018	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_886)	32	R/W	0000_0000h	16.2.13/ 754
101C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_887)	32	R/W	0000_0000h	16.2.13/ 754
1020	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_888)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1024	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_889)	32	R/W	0000_0000h	16.2.13/ 754
1028	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_890)	32	R/W	0000_0000h	16.2.13/ 754
102C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_891)	32	R/W	0000_0000h	16.2.13/ 754
1030	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_892)	32	R/W	0000_0000h	16.2.13/ 754
1034	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_893)	32	R/W	0000_0000h	16.2.13/ 754
1038	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_894)	32	R/W	0000_0000h	16.2.13/ 754
103C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_895)	32	R/W	0000_0000h	16.2.13/ 754
1040	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_896)	32	R/W	0000_0000h	16.2.13/ 754
1044	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_897)	32	R/W	0000_0000h	16.2.13/ 754
1048	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_898)	32	R/W	0000_0000h	16.2.13/ 754
104C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_899)	32	R/W	0000_0000h	16.2.13/ 754
1050	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_900)	32	R/W	0000_0000h	16.2.13/ 754
1054	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_901)	32	R/W	0000_0000h	16.2.13/ 754
1058	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_902)	32	R/W	0000_0000h	16.2.13/ 754
105C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_903)	32	R/W	0000_0000h	16.2.13/ 754
1060	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_904)	32	R/W	0000_0000h	16.2.13/ 754
1064	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_905)	32	R/W	0000_0000h	16.2.13/ 754
1068	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_906)	32	R/W	0000_0000h	16.2.13/ 754
106C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_907)	32	R/W	0000_0000h	16.2.13/ 754
1070	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_908)	32	R/W	0000_0000h	16.2.13/ 754
1074	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_909)	32	R/W	0000_0000h	16.2.13/ 754
1078	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_910)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
107C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_911)	32	R/W	0000_0000h	16.2.13/ 754
1080	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_912)	32	R/W	0000_0000h	16.2.13/ 754
1084	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_913)	32	R/W	0000_0000h	16.2.13/ 754
1088	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_914)	32	R/W	0000_0000h	16.2.13/ 754
108C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_915)	32	R/W	0000_0000h	16.2.13/ 754
1090	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_916)	32	R/W	0000_0000h	16.2.13/ 754
1094	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_917)	32	R/W	0000_0000h	16.2.13/ 754
1098	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_918)	32	R/W	0000_0000h	16.2.13/ 754
109C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_919)	32	R/W	0000_0000h	16.2.13/ 754
10A0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_920)	32	R/W	0000_0000h	16.2.13/ 754
10A4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_921)	32	R/W	0000_0000h	16.2.13/ 754
10A8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_922)	32	R/W	0000_0000h	16.2.13/ 754
10AC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_923)	32	R/W	0000_0000h	16.2.13/ 754
10B0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_924)	32	R/W	0000_0000h	16.2.13/ 754
10B4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_925)	32	R/W	0000_0000h	16.2.13/ 754
10B8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_926)	32	R/W	0000_0000h	16.2.13/ 754
10BC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_927)	32	R/W	0000_0000h	16.2.13/ 754
10C0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_928)	32	R/W	0000_0000h	16.2.13/ 754
10C4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_929)	32	R/W	0000_0000h	16.2.13/ 754
10C8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_930)	32	R/W	0000_0000h	16.2.13/ 754
10CC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_931)	32	R/W	0000_0000h	16.2.13/ 754
10D0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_932)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10D4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_933)	32	R/W	0000_0000h	16.2.13/ 754
10D8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_934)	32	R/W	0000_0000h	16.2.13/ 754
10DC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_935)	32	R/W	0000_0000h	16.2.13/ 754
10E0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_936)	32	R/W	0000_0000h	16.2.13/ 754
10E4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_937)	32	R/W	0000_0000h	16.2.13/ 754
10E8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_938)	32	R/W	0000_0000h	16.2.13/ 754
10EC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_939)	32	R/W	0000_0000h	16.2.13/ 754
10F0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_940)	32	R/W	0000_0000h	16.2.13/ 754
10F4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_941)	32	R/W	0000_0000h	16.2.13/ 754
10F8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_942)	32	R/W	0000_0000h	16.2.13/ 754
10FC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_943)	32	R/W	0000_0000h	16.2.13/ 754
1100	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_944)	32	R/W	0000_0000h	16.2.13/ 754
1104	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_945)	32	R/W	0000_0000h	16.2.13/ 754
1108	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_946)	32	R/W	0000_0000h	16.2.13/ 754
110C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_947)	32	R/W	0000_0000h	16.2.13/ 754
1110	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_948)	32	R/W	0000_0000h	16.2.13/ 754
1114	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_949)	32	R/W	0000_0000h	16.2.13/ 754
1118	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_950)	32	R/W	0000_0000h	16.2.13/ 754
111C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_951)	32	R/W	0000_0000h	16.2.13/ 754
1120	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_952)	32	R/W	0000_0000h	16.2.13/ 754
1124	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_953)	32	R/W	0000_0000h	16.2.13/ 754
1128	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_954)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
112C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_955)	32	R/W	0000_0000h	16.2.13/ 754
1130	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_956)	32	R/W	0000_0000h	16.2.13/ 754
1134	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_957)	32	R/W	0000_0000h	16.2.13/ 754
1138	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_958)	32	R/W	0000_0000h	16.2.13/ 754
113C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_959)	32	R/W	0000_0000h	16.2.13/ 754
1140	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_960)	32	R/W	0000_0000h	16.2.13/ 754
1144	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_961)	32	R/W	0000_0000h	16.2.13/ 754
1148	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_962)	32	R/W	0000_0000h	16.2.13/ 754
114C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_963)	32	R/W	0000_0000h	16.2.13/ 754
1150	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_964)	32	R/W	0000_0000h	16.2.13/ 754
1154	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_965)	32	R/W	0000_0000h	16.2.13/ 754
1158	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_966)	32	R/W	0000_0000h	16.2.13/ 754
115C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_967)	32	R/W	0000_0000h	16.2.13/ 754
1160	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_968)	32	R/W	0000_0000h	16.2.13/ 754
1164	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_969)	32	R/W	0000_0000h	16.2.13/ 754
1168	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_970)	32	R/W	0000_0000h	16.2.13/ 754
116C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_971)	32	R/W	0000_0000h	16.2.13/ 754
1170	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_972)	32	R/W	0000_0000h	16.2.13/ 754
1174	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_973)	32	R/W	0000_0000h	16.2.13/ 754
1178	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_974)	32	R/W	0000_0000h	16.2.13/ 754
117C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_975)	32	R/W	0000_0000h	16.2.13/ 754
1180	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_976)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1184	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_977)	32	R/W	0000_0000h	16.2.13/ 754
1188	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_978)	32	R/W	0000_0000h	16.2.13/ 754
118C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_979)	32	R/W	0000_0000h	16.2.13/ 754
1190	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_980)	32	R/W	0000_0000h	16.2.13/ 754
1194	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_981)	32	R/W	0000_0000h	16.2.13/ 754
1198	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_982)	32	R/W	0000_0000h	16.2.13/ 754
119C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_983)	32	R/W	0000_0000h	16.2.13/ 754
11A0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_984)	32	R/W	0000_0000h	16.2.13/ 754
11A4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_985)	32	R/W	0000_0000h	16.2.13/ 754
11A8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_986)	32	R/W	0000_0000h	16.2.13/ 754
11AC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_987)	32	R/W	0000_0000h	16.2.13/ 754
11B0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_988)	32	R/W	0000_0000h	16.2.13/ 754
11B4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_989)	32	R/W	0000_0000h	16.2.13/ 754
11B8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_990)	32	R/W	0000_0000h	16.2.13/ 754
11BC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_991)	32	R/W	0000_0000h	16.2.13/ 754
11C0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_992)	32	R/W	0000_0000h	16.2.13/ 754
11C4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_993)	32	R/W	0000_0000h	16.2.13/ 754
11C8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_994)	32	R/W	0000_0000h	16.2.13/ 754
11CC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_995)	32	R/W	0000_0000h	16.2.13/ 754
11D0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_996)	32	R/W	0000_0000h	16.2.13/ 754
11D4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_997)	32	R/W	0000_0000h	16.2.13/ 754
11D8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_998)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11DC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_999)	32	R/W	0000_0000h	16.2.13/ 754
11E0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1000)	32	R/W	0000_0000h	16.2.13/ 754
11E4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1001)	32	R/W	0000_0000h	16.2.13/ 754
11E8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1002)	32	R/W	0000_0000h	16.2.13/ 754
11EC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1003)	32	R/W	0000_0000h	16.2.13/ 754
11F0	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1004)	32	R/W	0000_0000h	16.2.13/ 754
11F4	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1005)	32	R/W	0000_0000h	16.2.13/ 754
11F8	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1006)	32	R/W	0000_0000h	16.2.13/ 754
11FC	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1007)	32	R/W	0000_0000h	16.2.13/ 754
1200	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1008)	32	R/W	0000_0000h	16.2.13/ 754
1204	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1009)	32	R/W	0000_0000h	16.2.13/ 754
1208	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1010)	32	R/W	0000_0000h	16.2.13/ 754
120C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1011)	32	R/W	0000_0000h	16.2.13/ 754
1210	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1012)	32	R/W	0000_0000h	16.2.13/ 754
1214	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1013)	32	R/W	0000_0000h	16.2.13/ 754
1218	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1014)	32	R/W	0000_0000h	16.2.13/ 754
121C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1015)	32	R/W	0000_0000h	16.2.13/ 754
1220	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1016)	32	R/W	0000_0000h	16.2.13/ 754
1224	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1017)	32	R/W	0000_0000h	16.2.13/ 754
1228	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1018)	32	R/W	0000_0000h	16.2.13/ 754
122C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1019)	32	R/W	0000_0000h	16.2.13/ 754
1230	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1020)	32	R/W	0000_0000h	16.2.13/ 754

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1234	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1021)	32	R/W	0000_0000h	16.2.13/ 754
1238	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1022)	32	R/W	0000_0000h	16.2.13/ 754
123C	Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_1023)	32	R/W	0000_0000h	16.2.13/ 754
1300	GPIO Pad Data Out Register (SIUL2_GPDO0)	8	R/W	00h	16.2.14/ 755
1301	GPIO Pad Data Out Register (SIUL2_GPDO1)	8	R/W	00h	16.2.14/ 755
1302	GPIO Pad Data Out Register (SIUL2_GPDO2)	8	R/W	00h	16.2.14/ 755
1303	GPIO Pad Data Out Register (SIUL2_GPDO3)	8	R/W	00h	16.2.14/ 755
1304	GPIO Pad Data Out Register (SIUL2_GPDO4)	8	R/W	00h	16.2.14/ 755
1305	GPIO Pad Data Out Register (SIUL2_GPDO5)	8	R/W	00h	16.2.14/ 755
1306	GPIO Pad Data Out Register (SIUL2_GPDO6)	8	R/W	00h	16.2.14/ 755
1307	GPIO Pad Data Out Register (SIUL2_GPDO7)	8	R/W	00h	16.2.14/ 755
1308	GPIO Pad Data Out Register (SIUL2_GPDO8)	8	R/W	00h	16.2.14/ 755
1309	GPIO Pad Data Out Register (SIUL2_GPDO9)	8	R/W	00h	16.2.14/ 755
130A	GPIO Pad Data Out Register (SIUL2_GPDO10)	8	R/W	00h	16.2.14/ 755
130B	GPIO Pad Data Out Register (SIUL2_GPDO11)	8	R/W	00h	16.2.14/ 755
130C	GPIO Pad Data Out Register (SIUL2_GPDO12)	8	R/W	00h	16.2.14/ 755
130D	GPIO Pad Data Out Register (SIUL2_GPDO13)	8	R/W	00h	16.2.14/ 755
130E	GPIO Pad Data Out Register (SIUL2_GPDO14)	8	R/W	00h	16.2.14/ 755
130F	GPIO Pad Data Out Register (SIUL2_GPDO15)	8	R/W	00h	16.2.14/ 755
1310	GPIO Pad Data Out Register (SIUL2_GPDO16)	8	R/W	00h	16.2.14/ 755
1311	GPIO Pad Data Out Register (SIUL2_GPDO17)	8	R/W	00h	16.2.14/ 755
1312	GPIO Pad Data Out Register (SIUL2_GPDO18)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1313	GPIO Pad Data Out Register (SIUL2_GPDO19)	8	R/W	00h	16.2.14/ 755
1314	GPIO Pad Data Out Register (SIUL2_GPDO20)	8	R/W	00h	16.2.14/ 755
1315	GPIO Pad Data Out Register (SIUL2_GPDO21)	8	R/W	00h	16.2.14/ 755
1316	GPIO Pad Data Out Register (SIUL2_GPDO22)	8	R/W	00h	16.2.14/ 755
1317	GPIO Pad Data Out Register (SIUL2_GPDO23)	8	R/W	00h	16.2.14/ 755
1318	GPIO Pad Data Out Register (SIUL2_GPDO24)	8	R/W	00h	16.2.14/ 755
1319	GPIO Pad Data Out Register (SIUL2_GPDO25)	8	R/W	00h	16.2.14/ 755
131A	GPIO Pad Data Out Register (SIUL2_GPDO26)	8	R/W	00h	16.2.14/ 755
131B	GPIO Pad Data Out Register (SIUL2_GPDO27)	8	R/W	00h	16.2.14/ 755
131C	GPIO Pad Data Out Register (SIUL2_GPDO28)	8	R/W	00h	16.2.14/ 755
131D	GPIO Pad Data Out Register (SIUL2_GPDO29)	8	R/W	00h	16.2.14/ 755
131E	GPIO Pad Data Out Register (SIUL2_GPDO30)	8	R/W	00h	16.2.14/ 755
131F	GPIO Pad Data Out Register (SIUL2_GPDO31)	8	R/W	00h	16.2.14/ 755
1320	GPIO Pad Data Out Register (SIUL2_GPDO32)	8	R/W	00h	16.2.14/ 755
1321	GPIO Pad Data Out Register (SIUL2_GPDO33)	8	R/W	00h	16.2.14/ 755
1322	GPIO Pad Data Out Register (SIUL2_GPDO34)	8	R/W	00h	16.2.14/ 755
1323	GPIO Pad Data Out Register (SIUL2_GPDO35)	8	R/W	00h	16.2.14/ 755
1324	GPIO Pad Data Out Register (SIUL2_GPDO36)	8	R/W	00h	16.2.14/ 755
1325	GPIO Pad Data Out Register (SIUL2_GPDO37)	8	R/W	00h	16.2.14/ 755
1326	GPIO Pad Data Out Register (SIUL2_GPDO38)	8	R/W	00h	16.2.14/ 755
1327	GPIO Pad Data Out Register (SIUL2_GPDO39)	8	R/W	00h	16.2.14/ 755
1328	GPIO Pad Data Out Register (SIUL2_GPDO40)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1329	GPIO Pad Data Out Register (SIUL2_GPDO41)	8	R/W	00h	16.2.14/ 755
132A	GPIO Pad Data Out Register (SIUL2_GPDO42)	8	R/W	00h	16.2.14/ 755
132B	GPIO Pad Data Out Register (SIUL2_GPDO43)	8	R/W	00h	16.2.14/ 755
132C	GPIO Pad Data Out Register (SIUL2_GPDO44)	8	R/W	00h	16.2.14/ 755
132D	GPIO Pad Data Out Register (SIUL2_GPDO45)	8	R/W	00h	16.2.14/ 755
132E	GPIO Pad Data Out Register (SIUL2_GPDO46)	8	R/W	00h	16.2.14/ 755
132F	GPIO Pad Data Out Register (SIUL2_GPDO47)	8	R/W	00h	16.2.14/ 755
1330	GPIO Pad Data Out Register (SIUL2_GPDO48)	8	R/W	00h	16.2.14/ 755
1331	GPIO Pad Data Out Register (SIUL2_GPDO49)	8	R/W	00h	16.2.14/ 755
1332	GPIO Pad Data Out Register (SIUL2_GPDO50)	8	R/W	00h	16.2.14/ 755
1333	GPIO Pad Data Out Register (SIUL2_GPDO51)	8	R/W	00h	16.2.14/ 755
1334	GPIO Pad Data Out Register (SIUL2_GPDO52)	8	R/W	00h	16.2.14/ 755
1335	GPIO Pad Data Out Register (SIUL2_GPDO53)	8	R/W	00h	16.2.14/ 755
1336	GPIO Pad Data Out Register (SIUL2_GPDO54)	8	R/W	00h	16.2.14/ 755
1337	GPIO Pad Data Out Register (SIUL2_GPDO55)	8	R/W	00h	16.2.14/ 755
1338	GPIO Pad Data Out Register (SIUL2_GPDO56)	8	R/W	00h	16.2.14/ 755
1339	GPIO Pad Data Out Register (SIUL2_GPDO57)	8	R/W	00h	16.2.14/ 755
133A	GPIO Pad Data Out Register (SIUL2_GPDO58)	8	R/W	00h	16.2.14/ 755
133B	GPIO Pad Data Out Register (SIUL2_GPDO59)	8	R/W	00h	16.2.14/ 755
133C	GPIO Pad Data Out Register (SIUL2_GPDO60)	8	R/W	00h	16.2.14/ 755
133D	GPIO Pad Data Out Register (SIUL2_GPDO61)	8	R/W	00h	16.2.14/ 755
133E	GPIO Pad Data Out Register (SIUL2_GPDO62)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
133F	GPIO Pad Data Out Register (SIUL2_GPDO63)	8	R/W	00h	16.2.14/ 755
1340	GPIO Pad Data Out Register (SIUL2_GPDO64)	8	R/W	00h	16.2.14/ 755
1341	GPIO Pad Data Out Register (SIUL2_GPDO65)	8	R/W	00h	16.2.14/ 755
1342	GPIO Pad Data Out Register (SIUL2_GPDO66)	8	R/W	00h	16.2.14/ 755
1343	GPIO Pad Data Out Register (SIUL2_GPDO67)	8	R/W	00h	16.2.14/ 755
1344	GPIO Pad Data Out Register (SIUL2_GPDO68)	8	R/W	00h	16.2.14/ 755
1345	GPIO Pad Data Out Register (SIUL2_GPDO69)	8	R/W	00h	16.2.14/ 755
1346	GPIO Pad Data Out Register (SIUL2_GPDO70)	8	R/W	00h	16.2.14/ 755
1347	GPIO Pad Data Out Register (SIUL2_GPDO71)	8	R/W	00h	16.2.14/ 755
1348	GPIO Pad Data Out Register (SIUL2_GPDO72)	8	R/W	00h	16.2.14/ 755
1349	GPIO Pad Data Out Register (SIUL2_GPDO73)	8	R/W	00h	16.2.14/ 755
134A	GPIO Pad Data Out Register (SIUL2_GPDO74)	8	R/W	00h	16.2.14/ 755
134B	GPIO Pad Data Out Register (SIUL2_GPDO75)	8	R/W	00h	16.2.14/ 755
134C	GPIO Pad Data Out Register (SIUL2_GPDO76)	8	R/W	00h	16.2.14/ 755
134D	GPIO Pad Data Out Register (SIUL2_GPDO77)	8	R/W	00h	16.2.14/ 755
134E	GPIO Pad Data Out Register (SIUL2_GPDO78)	8	R/W	00h	16.2.14/ 755
134F	GPIO Pad Data Out Register (SIUL2_GPDO79)	8	R/W	00h	16.2.14/ 755
1350	GPIO Pad Data Out Register (SIUL2_GPDO80)	8	R/W	00h	16.2.14/ 755
1351	GPIO Pad Data Out Register (SIUL2_GPDO81)	8	R/W	00h	16.2.14/ 755
1352	GPIO Pad Data Out Register (SIUL2_GPDO82)	8	R/W	00h	16.2.14/ 755
1353	GPIO Pad Data Out Register (SIUL2_GPDO83)	8	R/W	00h	16.2.14/ 755
1354	GPIO Pad Data Out Register (SIUL2_GPDO84)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1355	GPIO Pad Data Out Register (SIUL2_GPDO85)	8	R/W	00h	16.2.14/ 755
1356	GPIO Pad Data Out Register (SIUL2_GPDO86)	8	R/W	00h	16.2.14/ 755
1357	GPIO Pad Data Out Register (SIUL2_GPDO87)	8	R/W	00h	16.2.14/ 755
1358	GPIO Pad Data Out Register (SIUL2_GPDO88)	8	R/W	00h	16.2.14/ 755
1359	GPIO Pad Data Out Register (SIUL2_GPDO89)	8	R/W	00h	16.2.14/ 755
135A	GPIO Pad Data Out Register (SIUL2_GPDO90)	8	R/W	00h	16.2.14/ 755
135B	GPIO Pad Data Out Register (SIUL2_GPDO91)	8	R/W	00h	16.2.14/ 755
135C	GPIO Pad Data Out Register (SIUL2_GPDO92)	8	R/W	00h	16.2.14/ 755
135D	GPIO Pad Data Out Register (SIUL2_GPDO93)	8	R/W	00h	16.2.14/ 755
135E	GPIO Pad Data Out Register (SIUL2_GPDO94)	8	R/W	00h	16.2.14/ 755
135F	GPIO Pad Data Out Register (SIUL2_GPDO95)	8	R/W	00h	16.2.14/ 755
1360	GPIO Pad Data Out Register (SIUL2_GPDO96)	8	R/W	00h	16.2.14/ 755
1361	GPIO Pad Data Out Register (SIUL2_GPDO97)	8	R/W	00h	16.2.14/ 755
1362	GPIO Pad Data Out Register (SIUL2_GPDO98)	8	R/W	00h	16.2.14/ 755
1363	GPIO Pad Data Out Register (SIUL2_GPDO99)	8	R/W	00h	16.2.14/ 755
1364	GPIO Pad Data Out Register (SIUL2_GPDO100)	8	R/W	00h	16.2.14/ 755
1365	GPIO Pad Data Out Register (SIUL2_GPDO101)	8	R/W	00h	16.2.14/ 755
1366	GPIO Pad Data Out Register (SIUL2_GPDO102)	8	R/W	00h	16.2.14/ 755
1367	GPIO Pad Data Out Register (SIUL2_GPDO103)	8	R/W	00h	16.2.14/ 755
1368	GPIO Pad Data Out Register (SIUL2_GPDO104)	8	R/W	00h	16.2.14/ 755
1369	GPIO Pad Data Out Register (SIUL2_GPDO105)	8	R/W	00h	16.2.14/ 755
136A	GPIO Pad Data Out Register (SIUL2_GPDO106)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
136B	GPIO Pad Data Out Register (SIUL2_GPDO107)	8	R/W	00h	16.2.14/ 755
136C	GPIO Pad Data Out Register (SIUL2_GPDO108)	8	R/W	00h	16.2.14/ 755
136D	GPIO Pad Data Out Register (SIUL2_GPDO109)	8	R/W	00h	16.2.14/ 755
136E	GPIO Pad Data Out Register (SIUL2_GPDO110)	8	R/W	00h	16.2.14/ 755
136F	GPIO Pad Data Out Register (SIUL2_GPDO111)	8	R/W	00h	16.2.14/ 755
1370	GPIO Pad Data Out Register (SIUL2_GPDO112)	8	R/W	00h	16.2.14/ 755
1371	GPIO Pad Data Out Register (SIUL2_GPDO113)	8	R/W	00h	16.2.14/ 755
1372	GPIO Pad Data Out Register (SIUL2_GPDO114)	8	R/W	00h	16.2.14/ 755
1373	GPIO Pad Data Out Register (SIUL2_GPDO115)	8	R/W	00h	16.2.14/ 755
1374	GPIO Pad Data Out Register (SIUL2_GPDO116)	8	R/W	00h	16.2.14/ 755
1375	GPIO Pad Data Out Register (SIUL2_GPDO117)	8	R/W	00h	16.2.14/ 755
1376	GPIO Pad Data Out Register (SIUL2_GPDO118)	8	R/W	00h	16.2.14/ 755
1377	GPIO Pad Data Out Register (SIUL2_GPDO119)	8	R/W	00h	16.2.14/ 755
1378	GPIO Pad Data Out Register (SIUL2_GPDO120)	8	R/W	00h	16.2.14/ 755
1379	GPIO Pad Data Out Register (SIUL2_GPDO121)	8	R/W	00h	16.2.14/ 755
137A	GPIO Pad Data Out Register (SIUL2_GPDO122)	8	R/W	00h	16.2.14/ 755
137B	GPIO Pad Data Out Register (SIUL2_GPDO123)	8	R/W	00h	16.2.14/ 755
137C	GPIO Pad Data Out Register (SIUL2_GPDO124)	8	R/W	00h	16.2.14/ 755
137D	GPIO Pad Data Out Register (SIUL2_GPDO125)	8	R/W	00h	16.2.14/ 755
137E	GPIO Pad Data Out Register (SIUL2_GPDO126)	8	R/W	00h	16.2.14/ 755
137F	GPIO Pad Data Out Register (SIUL2_GPDO127)	8	R/W	00h	16.2.14/ 755
1380	GPIO Pad Data Out Register (SIUL2_GPDO128)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1381	GPIO Pad Data Out Register (SIUL2_GPDO129)	8	R/W	00h	16.2.14/ 755
1382	GPIO Pad Data Out Register (SIUL2_GPDO130)	8	R/W	00h	16.2.14/ 755
1383	GPIO Pad Data Out Register (SIUL2_GPDO131)	8	R/W	00h	16.2.14/ 755
1384	GPIO Pad Data Out Register (SIUL2_GPDO132)	8	R/W	00h	16.2.14/ 755
1385	GPIO Pad Data Out Register (SIUL2_GPDO133)	8	R/W	00h	16.2.14/ 755
1386	GPIO Pad Data Out Register (SIUL2_GPDO134)	8	R/W	00h	16.2.14/ 755
1387	GPIO Pad Data Out Register (SIUL2_GPDO135)	8	R/W	00h	16.2.14/ 755
1388	GPIO Pad Data Out Register (SIUL2_GPDO136)	8	R/W	00h	16.2.14/ 755
1389	GPIO Pad Data Out Register (SIUL2_GPDO137)	8	R/W	00h	16.2.14/ 755
138A	GPIO Pad Data Out Register (SIUL2_GPDO138)	8	R/W	00h	16.2.14/ 755
138B	GPIO Pad Data Out Register (SIUL2_GPDO139)	8	R/W	00h	16.2.14/ 755
138C	GPIO Pad Data Out Register (SIUL2_GPDO140)	8	R/W	00h	16.2.14/ 755
138D	GPIO Pad Data Out Register (SIUL2_GPDO141)	8	R/W	00h	16.2.14/ 755
138E	GPIO Pad Data Out Register (SIUL2_GPDO142)	8	R/W	00h	16.2.14/ 755
138F	GPIO Pad Data Out Register (SIUL2_GPDO143)	8	R/W	00h	16.2.14/ 755
1390	GPIO Pad Data Out Register (SIUL2_GPDO144)	8	R/W	00h	16.2.14/ 755
1391	GPIO Pad Data Out Register (SIUL2_GPDO145)	8	R/W	00h	16.2.14/ 755
1392	GPIO Pad Data Out Register (SIUL2_GPDO146)	8	R/W	00h	16.2.14/ 755
1393	GPIO Pad Data Out Register (SIUL2_GPDO147)	8	R/W	00h	16.2.14/ 755
1394	GPIO Pad Data Out Register (SIUL2_GPDO148)	8	R/W	00h	16.2.14/ 755
1395	GPIO Pad Data Out Register (SIUL2_GPDO149)	8	R/W	00h	16.2.14/ 755
1396	GPIO Pad Data Out Register (SIUL2_GPDO150)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1397	GPIO Pad Data Out Register (SIUL2_GPDO151)	8	R/W	00h	16.2.14/ 755
1398	GPIO Pad Data Out Register (SIUL2_GPDO152)	8	R/W	00h	16.2.14/ 755
1399	GPIO Pad Data Out Register (SIUL2_GPDO153)	8	R/W	00h	16.2.14/ 755
139A	GPIO Pad Data Out Register (SIUL2_GPDO154)	8	R/W	00h	16.2.14/ 755
139B	GPIO Pad Data Out Register (SIUL2_GPDO155)	8	R/W	00h	16.2.14/ 755
139C	GPIO Pad Data Out Register (SIUL2_GPDO156)	8	R/W	00h	16.2.14/ 755
139D	GPIO Pad Data Out Register (SIUL2_GPDO157)	8	R/W	00h	16.2.14/ 755
139E	GPIO Pad Data Out Register (SIUL2_GPDO158)	8	R/W	00h	16.2.14/ 755
139F	GPIO Pad Data Out Register (SIUL2_GPDO159)	8	R/W	00h	16.2.14/ 755
13A0	GPIO Pad Data Out Register (SIUL2_GPDO160)	8	R/W	00h	16.2.14/ 755
13A1	GPIO Pad Data Out Register (SIUL2_GPDO161)	8	R/W	00h	16.2.14/ 755
13A2	GPIO Pad Data Out Register (SIUL2_GPDO162)	8	R/W	00h	16.2.14/ 755
13A3	GPIO Pad Data Out Register (SIUL2_GPDO163)	8	R/W	00h	16.2.14/ 755
13A4	GPIO Pad Data Out Register (SIUL2_GPDO164)	8	R/W	00h	16.2.14/ 755
13A5	GPIO Pad Data Out Register (SIUL2_GPDO165)	8	R/W	00h	16.2.14/ 755
13A6	GPIO Pad Data Out Register (SIUL2_GPDO166)	8	R/W	00h	16.2.14/ 755
13A7	GPIO Pad Data Out Register (SIUL2_GPDO167)	8	R/W	00h	16.2.14/ 755
13A8	GPIO Pad Data Out Register (SIUL2_GPDO168)	8	R/W	00h	16.2.14/ 755
13A9	GPIO Pad Data Out Register (SIUL2_GPDO169)	8	R/W	00h	16.2.14/ 755
13AA	GPIO Pad Data Out Register (SIUL2_GPDO170)	8	R/W	00h	16.2.14/ 755
13AB	GPIO Pad Data Out Register (SIUL2_GPDO171)	8	R/W	00h	16.2.14/ 755
13AC	GPIO Pad Data Out Register (SIUL2_GPDO172)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13AD	GPIO Pad Data Out Register (SIUL2_GPDO173)	8	R/W	00h	16.2.14/ 755
13AE	GPIO Pad Data Out Register (SIUL2_GPDO174)	8	R/W	00h	16.2.14/ 755
13AF	GPIO Pad Data Out Register (SIUL2_GPDO175)	8	R/W	00h	16.2.14/ 755
13B0	GPIO Pad Data Out Register (SIUL2_GPDO176)	8	R/W	00h	16.2.14/ 755
13B1	GPIO Pad Data Out Register (SIUL2_GPDO177)	8	R/W	00h	16.2.14/ 755
13B2	GPIO Pad Data Out Register (SIUL2_GPDO178)	8	R/W	00h	16.2.14/ 755
13B3	GPIO Pad Data Out Register (SIUL2_GPDO179)	8	R/W	00h	16.2.14/ 755
13B4	GPIO Pad Data Out Register (SIUL2_GPDO180)	8	R/W	00h	16.2.14/ 755
13B5	GPIO Pad Data Out Register (SIUL2_GPDO181)	8	R/W	00h	16.2.14/ 755
13B6	GPIO Pad Data Out Register (SIUL2_GPDO182)	8	R/W	00h	16.2.14/ 755
13B7	GPIO Pad Data Out Register (SIUL2_GPDO183)	8	R/W	00h	16.2.14/ 755
13B8	GPIO Pad Data Out Register (SIUL2_GPDO184)	8	R/W	00h	16.2.14/ 755
13B9	GPIO Pad Data Out Register (SIUL2_GPDO185)	8	R/W	00h	16.2.14/ 755
13BA	GPIO Pad Data Out Register (SIUL2_GPDO186)	8	R/W	00h	16.2.14/ 755
13BB	GPIO Pad Data Out Register (SIUL2_GPDO187)	8	R/W	00h	16.2.14/ 755
13BC	GPIO Pad Data Out Register (SIUL2_GPDO188)	8	R/W	00h	16.2.14/ 755
13BD	GPIO Pad Data Out Register (SIUL2_GPDO189)	8	R/W	00h	16.2.14/ 755
13BE	GPIO Pad Data Out Register (SIUL2_GPDO190)	8	R/W	00h	16.2.14/ 755
13BF	GPIO Pad Data Out Register (SIUL2_GPDO191)	8	R/W	00h	16.2.14/ 755
13C0	GPIO Pad Data Out Register (SIUL2_GPDO192)	8	R/W	00h	16.2.14/ 755
13C1	GPIO Pad Data Out Register (SIUL2_GPDO193)	8	R/W	00h	16.2.14/ 755
13C2	GPIO Pad Data Out Register (SIUL2_GPDO194)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13C3	GPIO Pad Data Out Register (SIUL2_GPDO195)	8	R/W	00h	16.2.14/ 755
13C4	GPIO Pad Data Out Register (SIUL2_GPDO196)	8	R/W	00h	16.2.14/ 755
13C5	GPIO Pad Data Out Register (SIUL2_GPDO197)	8	R/W	00h	16.2.14/ 755
13C6	GPIO Pad Data Out Register (SIUL2_GPDO198)	8	R/W	00h	16.2.14/ 755
13C7	GPIO Pad Data Out Register (SIUL2_GPDO199)	8	R/W	00h	16.2.14/ 755
13C8	GPIO Pad Data Out Register (SIUL2_GPDO200)	8	R/W	00h	16.2.14/ 755
13C9	GPIO Pad Data Out Register (SIUL2_GPDO201)	8	R/W	00h	16.2.14/ 755
13CA	GPIO Pad Data Out Register (SIUL2_GPDO202)	8	R/W	00h	16.2.14/ 755
13CB	GPIO Pad Data Out Register (SIUL2_GPDO203)	8	R/W	00h	16.2.14/ 755
13CC	GPIO Pad Data Out Register (SIUL2_GPDO204)	8	R/W	00h	16.2.14/ 755
13CD	GPIO Pad Data Out Register (SIUL2_GPDO205)	8	R/W	00h	16.2.14/ 755
13CE	GPIO Pad Data Out Register (SIUL2_GPDO206)	8	R/W	00h	16.2.14/ 755
13CF	GPIO Pad Data Out Register (SIUL2_GPDO207)	8	R/W	00h	16.2.14/ 755
13D0	GPIO Pad Data Out Register (SIUL2_GPDO208)	8	R/W	00h	16.2.14/ 755
13D1	GPIO Pad Data Out Register (SIUL2_GPDO209)	8	R/W	00h	16.2.14/ 755
13D2	GPIO Pad Data Out Register (SIUL2_GPDO210)	8	R/W	00h	16.2.14/ 755
13D3	GPIO Pad Data Out Register (SIUL2_GPDO211)	8	R/W	00h	16.2.14/ 755
13D4	GPIO Pad Data Out Register (SIUL2_GPDO212)	8	R/W	00h	16.2.14/ 755
13D5	GPIO Pad Data Out Register (SIUL2_GPDO213)	8	R/W	00h	16.2.14/ 755
13D6	GPIO Pad Data Out Register (SIUL2_GPDO214)	8	R/W	00h	16.2.14/ 755
13D7	GPIO Pad Data Out Register (SIUL2_GPDO215)	8	R/W	00h	16.2.14/ 755
13D8	GPIO Pad Data Out Register (SIUL2_GPDO216)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13D9	GPIO Pad Data Out Register (SIUL2_GPDO217)	8	R/W	00h	16.2.14/ 755
13DA	GPIO Pad Data Out Register (SIUL2_GPDO218)	8	R/W	00h	16.2.14/ 755
13DB	GPIO Pad Data Out Register (SIUL2_GPDO219)	8	R/W	00h	16.2.14/ 755
13DC	GPIO Pad Data Out Register (SIUL2_GPDO220)	8	R/W	00h	16.2.14/ 755
13DD	GPIO Pad Data Out Register (SIUL2_GPDO221)	8	R/W	00h	16.2.14/ 755
13DE	GPIO Pad Data Out Register (SIUL2_GPDO222)	8	R/W	00h	16.2.14/ 755
13DF	GPIO Pad Data Out Register (SIUL2_GPDO223)	8	R/W	00h	16.2.14/ 755
13E0	GPIO Pad Data Out Register (SIUL2_GPDO224)	8	R/W	00h	16.2.14/ 755
13E1	GPIO Pad Data Out Register (SIUL2_GPDO225)	8	R/W	00h	16.2.14/ 755
13E2	GPIO Pad Data Out Register (SIUL2_GPDO226)	8	R/W	00h	16.2.14/ 755
13E3	GPIO Pad Data Out Register (SIUL2_GPDO227)	8	R/W	00h	16.2.14/ 755
13E4	GPIO Pad Data Out Register (SIUL2_GPDO228)	8	R/W	00h	16.2.14/ 755
13E5	GPIO Pad Data Out Register (SIUL2_GPDO229)	8	R/W	00h	16.2.14/ 755
13E6	GPIO Pad Data Out Register (SIUL2_GPDO230)	8	R/W	00h	16.2.14/ 755
13E7	GPIO Pad Data Out Register (SIUL2_GPDO231)	8	R/W	00h	16.2.14/ 755
13E8	GPIO Pad Data Out Register (SIUL2_GPDO232)	8	R/W	00h	16.2.14/ 755
13E9	GPIO Pad Data Out Register (SIUL2_GPDO233)	8	R/W	00h	16.2.14/ 755
13EA	GPIO Pad Data Out Register (SIUL2_GPDO234)	8	R/W	00h	16.2.14/ 755
13EB	GPIO Pad Data Out Register (SIUL2_GPDO235)	8	R/W	00h	16.2.14/ 755
13EC	GPIO Pad Data Out Register (SIUL2_GPDO236)	8	R/W	00h	16.2.14/ 755
13ED	GPIO Pad Data Out Register (SIUL2_GPDO237)	8	R/W	00h	16.2.14/ 755
13EE	GPIO Pad Data Out Register (SIUL2_GPDO238)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13EF	GPIO Pad Data Out Register (SIUL2_GPDO239)	8	R/W	00h	16.2.14/ 755
13F0	GPIO Pad Data Out Register (SIUL2_GPDO240)	8	R/W	00h	16.2.14/ 755
13F1	GPIO Pad Data Out Register (SIUL2_GPDO241)	8	R/W	00h	16.2.14/ 755
13F2	GPIO Pad Data Out Register (SIUL2_GPDO242)	8	R/W	00h	16.2.14/ 755
13F3	GPIO Pad Data Out Register (SIUL2_GPDO243)	8	R/W	00h	16.2.14/ 755
13F4	GPIO Pad Data Out Register (SIUL2_GPDO244)	8	R/W	00h	16.2.14/ 755
13F5	GPIO Pad Data Out Register (SIUL2_GPDO245)	8	R/W	00h	16.2.14/ 755
13F6	GPIO Pad Data Out Register (SIUL2_GPDO246)	8	R/W	00h	16.2.14/ 755
13F7	GPIO Pad Data Out Register (SIUL2_GPDO247)	8	R/W	00h	16.2.14/ 755
13F8	GPIO Pad Data Out Register (SIUL2_GPDO248)	8	R/W	00h	16.2.14/ 755
13F9	GPIO Pad Data Out Register (SIUL2_GPDO249)	8	R/W	00h	16.2.14/ 755
13FA	GPIO Pad Data Out Register (SIUL2_GPDO250)	8	R/W	00h	16.2.14/ 755
13FB	GPIO Pad Data Out Register (SIUL2_GPDO251)	8	R/W	00h	16.2.14/ 755
13FC	GPIO Pad Data Out Register (SIUL2_GPDO252)	8	R/W	00h	16.2.14/ 755
13FD	GPIO Pad Data Out Register (SIUL2_GPDO253)	8	R/W	00h	16.2.14/ 755
13FE	GPIO Pad Data Out Register (SIUL2_GPDO254)	8	R/W	00h	16.2.14/ 755
13FF	GPIO Pad Data Out Register (SIUL2_GPDO255)	8	R/W	00h	16.2.14/ 755
1400	GPIO Pad Data Out Register (SIUL2_GPDO256)	8	R/W	00h	16.2.14/ 755
1401	GPIO Pad Data Out Register (SIUL2_GPDO257)	8	R/W	00h	16.2.14/ 755
1402	GPIO Pad Data Out Register (SIUL2_GPDO258)	8	R/W	00h	16.2.14/ 755
1403	GPIO Pad Data Out Register (SIUL2_GPDO259)	8	R/W	00h	16.2.14/ 755
1404	GPIO Pad Data Out Register (SIUL2_GPDO260)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1405	GPIO Pad Data Out Register (SIUL2_GPDO261)	8	R/W	00h	16.2.14/ 755
1406	GPIO Pad Data Out Register (SIUL2_GPDO262)	8	R/W	00h	16.2.14/ 755
1407	GPIO Pad Data Out Register (SIUL2_GPDO263)	8	R/W	00h	16.2.14/ 755
1408	GPIO Pad Data Out Register (SIUL2_GPDO264)	8	R/W	00h	16.2.14/ 755
1409	GPIO Pad Data Out Register (SIUL2_GPDO265)	8	R/W	00h	16.2.14/ 755
140A	GPIO Pad Data Out Register (SIUL2_GPDO266)	8	R/W	00h	16.2.14/ 755
140B	GPIO Pad Data Out Register (SIUL2_GPDO267)	8	R/W	00h	16.2.14/ 755
140C	GPIO Pad Data Out Register (SIUL2_GPDO268)	8	R/W	00h	16.2.14/ 755
140D	GPIO Pad Data Out Register (SIUL2_GPDO269)	8	R/W	00h	16.2.14/ 755
140E	GPIO Pad Data Out Register (SIUL2_GPDO270)	8	R/W	00h	16.2.14/ 755
140F	GPIO Pad Data Out Register (SIUL2_GPDO271)	8	R/W	00h	16.2.14/ 755
1410	GPIO Pad Data Out Register (SIUL2_GPDO272)	8	R/W	00h	16.2.14/ 755
1411	GPIO Pad Data Out Register (SIUL2_GPDO273)	8	R/W	00h	16.2.14/ 755
1412	GPIO Pad Data Out Register (SIUL2_GPDO274)	8	R/W	00h	16.2.14/ 755
1413	GPIO Pad Data Out Register (SIUL2_GPDO275)	8	R/W	00h	16.2.14/ 755
1414	GPIO Pad Data Out Register (SIUL2_GPDO276)	8	R/W	00h	16.2.14/ 755
1415	GPIO Pad Data Out Register (SIUL2_GPDO277)	8	R/W	00h	16.2.14/ 755
1416	GPIO Pad Data Out Register (SIUL2_GPDO278)	8	R/W	00h	16.2.14/ 755
1417	GPIO Pad Data Out Register (SIUL2_GPDO279)	8	R/W	00h	16.2.14/ 755
1418	GPIO Pad Data Out Register (SIUL2_GPDO280)	8	R/W	00h	16.2.14/ 755
1419	GPIO Pad Data Out Register (SIUL2_GPDO281)	8	R/W	00h	16.2.14/ 755
141A	GPIO Pad Data Out Register (SIUL2_GPDO282)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
141B	GPIO Pad Data Out Register (SIUL2_GPDO283)	8	R/W	00h	16.2.14/ 755
141C	GPIO Pad Data Out Register (SIUL2_GPDO284)	8	R/W	00h	16.2.14/ 755
141D	GPIO Pad Data Out Register (SIUL2_GPDO285)	8	R/W	00h	16.2.14/ 755
141E	GPIO Pad Data Out Register (SIUL2_GPDO286)	8	R/W	00h	16.2.14/ 755
141F	GPIO Pad Data Out Register (SIUL2_GPDO287)	8	R/W	00h	16.2.14/ 755
1420	GPIO Pad Data Out Register (SIUL2_GPDO288)	8	R/W	00h	16.2.14/ 755
1421	GPIO Pad Data Out Register (SIUL2_GPDO289)	8	R/W	00h	16.2.14/ 755
1422	GPIO Pad Data Out Register (SIUL2_GPDO290)	8	R/W	00h	16.2.14/ 755
1423	GPIO Pad Data Out Register (SIUL2_GPDO291)	8	R/W	00h	16.2.14/ 755
1424	GPIO Pad Data Out Register (SIUL2_GPDO292)	8	R/W	00h	16.2.14/ 755
1425	GPIO Pad Data Out Register (SIUL2_GPDO293)	8	R/W	00h	16.2.14/ 755
1426	GPIO Pad Data Out Register (SIUL2_GPDO294)	8	R/W	00h	16.2.14/ 755
1427	GPIO Pad Data Out Register (SIUL2_GPDO295)	8	R/W	00h	16.2.14/ 755
1428	GPIO Pad Data Out Register (SIUL2_GPDO296)	8	R/W	00h	16.2.14/ 755
1429	GPIO Pad Data Out Register (SIUL2_GPDO297)	8	R/W	00h	16.2.14/ 755
142A	GPIO Pad Data Out Register (SIUL2_GPDO298)	8	R/W	00h	16.2.14/ 755
142B	GPIO Pad Data Out Register (SIUL2_GPDO299)	8	R/W	00h	16.2.14/ 755
142C	GPIO Pad Data Out Register (SIUL2_GPDO300)	8	R/W	00h	16.2.14/ 755
142D	GPIO Pad Data Out Register (SIUL2_GPDO301)	8	R/W	00h	16.2.14/ 755
142E	GPIO Pad Data Out Register (SIUL2_GPDO302)	8	R/W	00h	16.2.14/ 755
142F	GPIO Pad Data Out Register (SIUL2_GPDO303)	8	R/W	00h	16.2.14/ 755
1430	GPIO Pad Data Out Register (SIUL2_GPDO304)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1431	GPIO Pad Data Out Register (SIUL2_GPDO305)	8	R/W	00h	16.2.14/ 755
1432	GPIO Pad Data Out Register (SIUL2_GPDO306)	8	R/W	00h	16.2.14/ 755
1433	GPIO Pad Data Out Register (SIUL2_GPDO307)	8	R/W	00h	16.2.14/ 755
1434	GPIO Pad Data Out Register (SIUL2_GPDO308)	8	R/W	00h	16.2.14/ 755
1435	GPIO Pad Data Out Register (SIUL2_GPDO309)	8	R/W	00h	16.2.14/ 755
1436	GPIO Pad Data Out Register (SIUL2_GPDO310)	8	R/W	00h	16.2.14/ 755
1437	GPIO Pad Data Out Register (SIUL2_GPDO311)	8	R/W	00h	16.2.14/ 755
1438	GPIO Pad Data Out Register (SIUL2_GPDO312)	8	R/W	00h	16.2.14/ 755
1439	GPIO Pad Data Out Register (SIUL2_GPDO313)	8	R/W	00h	16.2.14/ 755
143A	GPIO Pad Data Out Register (SIUL2_GPDO314)	8	R/W	00h	16.2.14/ 755
143B	GPIO Pad Data Out Register (SIUL2_GPDO315)	8	R/W	00h	16.2.14/ 755
143C	GPIO Pad Data Out Register (SIUL2_GPDO316)	8	R/W	00h	16.2.14/ 755
143D	GPIO Pad Data Out Register (SIUL2_GPDO317)	8	R/W	00h	16.2.14/ 755
143E	GPIO Pad Data Out Register (SIUL2_GPDO318)	8	R/W	00h	16.2.14/ 755
143F	GPIO Pad Data Out Register (SIUL2_GPDO319)	8	R/W	00h	16.2.14/ 755
1440	GPIO Pad Data Out Register (SIUL2_GPDO320)	8	R/W	00h	16.2.14/ 755
1441	GPIO Pad Data Out Register (SIUL2_GPDO321)	8	R/W	00h	16.2.14/ 755
1442	GPIO Pad Data Out Register (SIUL2_GPDO322)	8	R/W	00h	16.2.14/ 755
1443	GPIO Pad Data Out Register (SIUL2_GPDO323)	8	R/W	00h	16.2.14/ 755
1444	GPIO Pad Data Out Register (SIUL2_GPDO324)	8	R/W	00h	16.2.14/ 755
1445	GPIO Pad Data Out Register (SIUL2_GPDO325)	8	R/W	00h	16.2.14/ 755
1446	GPIO Pad Data Out Register (SIUL2_GPDO326)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1447	GPIO Pad Data Out Register (SIUL2_GPDO327)	8	R/W	00h	16.2.14/ 755
1448	GPIO Pad Data Out Register (SIUL2_GPDO328)	8	R/W	00h	16.2.14/ 755
1449	GPIO Pad Data Out Register (SIUL2_GPDO329)	8	R/W	00h	16.2.14/ 755
144A	GPIO Pad Data Out Register (SIUL2_GPDO330)	8	R/W	00h	16.2.14/ 755
144B	GPIO Pad Data Out Register (SIUL2_GPDO331)	8	R/W	00h	16.2.14/ 755
144C	GPIO Pad Data Out Register (SIUL2_GPDO332)	8	R/W	00h	16.2.14/ 755
144D	GPIO Pad Data Out Register (SIUL2_GPDO333)	8	R/W	00h	16.2.14/ 755
144E	GPIO Pad Data Out Register (SIUL2_GPDO334)	8	R/W	00h	16.2.14/ 755
144F	GPIO Pad Data Out Register (SIUL2_GPDO335)	8	R/W	00h	16.2.14/ 755
1450	GPIO Pad Data Out Register (SIUL2_GPDO336)	8	R/W	00h	16.2.14/ 755
1451	GPIO Pad Data Out Register (SIUL2_GPDO337)	8	R/W	00h	16.2.14/ 755
1452	GPIO Pad Data Out Register (SIUL2_GPDO338)	8	R/W	00h	16.2.14/ 755
1453	GPIO Pad Data Out Register (SIUL2_GPDO339)	8	R/W	00h	16.2.14/ 755
1454	GPIO Pad Data Out Register (SIUL2_GPDO340)	8	R/W	00h	16.2.14/ 755
1455	GPIO Pad Data Out Register (SIUL2_GPDO341)	8	R/W	00h	16.2.14/ 755
1456	GPIO Pad Data Out Register (SIUL2_GPDO342)	8	R/W	00h	16.2.14/ 755
1457	GPIO Pad Data Out Register (SIUL2_GPDO343)	8	R/W	00h	16.2.14/ 755
1458	GPIO Pad Data Out Register (SIUL2_GPDO344)	8	R/W	00h	16.2.14/ 755
1459	GPIO Pad Data Out Register (SIUL2_GPDO345)	8	R/W	00h	16.2.14/ 755
145A	GPIO Pad Data Out Register (SIUL2_GPDO346)	8	R/W	00h	16.2.14/ 755
145B	GPIO Pad Data Out Register (SIUL2_GPDO347)	8	R/W	00h	16.2.14/ 755
145C	GPIO Pad Data Out Register (SIUL2_GPDO348)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
145D	GPIO Pad Data Out Register (SIUL2_GPDO349)	8	R/W	00h	16.2.14/ 755
145E	GPIO Pad Data Out Register (SIUL2_GPDO350)	8	R/W	00h	16.2.14/ 755
145F	GPIO Pad Data Out Register (SIUL2_GPDO351)	8	R/W	00h	16.2.14/ 755
1460	GPIO Pad Data Out Register (SIUL2_GPDO352)	8	R/W	00h	16.2.14/ 755
1461	GPIO Pad Data Out Register (SIUL2_GPDO353)	8	R/W	00h	16.2.14/ 755
1462	GPIO Pad Data Out Register (SIUL2_GPDO354)	8	R/W	00h	16.2.14/ 755
1463	GPIO Pad Data Out Register (SIUL2_GPDO355)	8	R/W	00h	16.2.14/ 755
1464	GPIO Pad Data Out Register (SIUL2_GPDO356)	8	R/W	00h	16.2.14/ 755
1465	GPIO Pad Data Out Register (SIUL2_GPDO357)	8	R/W	00h	16.2.14/ 755
1466	GPIO Pad Data Out Register (SIUL2_GPDO358)	8	R/W	00h	16.2.14/ 755
1467	GPIO Pad Data Out Register (SIUL2_GPDO359)	8	R/W	00h	16.2.14/ 755
1468	GPIO Pad Data Out Register (SIUL2_GPDO360)	8	R/W	00h	16.2.14/ 755
1469	GPIO Pad Data Out Register (SIUL2_GPDO361)	8	R/W	00h	16.2.14/ 755
146A	GPIO Pad Data Out Register (SIUL2_GPDO362)	8	R/W	00h	16.2.14/ 755
146B	GPIO Pad Data Out Register (SIUL2_GPDO363)	8	R/W	00h	16.2.14/ 755
146C	GPIO Pad Data Out Register (SIUL2_GPDO364)	8	R/W	00h	16.2.14/ 755
146D	GPIO Pad Data Out Register (SIUL2_GPDO365)	8	R/W	00h	16.2.14/ 755
146E	GPIO Pad Data Out Register (SIUL2_GPDO366)	8	R/W	00h	16.2.14/ 755
146F	GPIO Pad Data Out Register (SIUL2_GPDO367)	8	R/W	00h	16.2.14/ 755
1470	GPIO Pad Data Out Register (SIUL2_GPDO368)	8	R/W	00h	16.2.14/ 755
1471	GPIO Pad Data Out Register (SIUL2_GPDO369)	8	R/W	00h	16.2.14/ 755
1472	GPIO Pad Data Out Register (SIUL2_GPDO370)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1473	GPIO Pad Data Out Register (SIUL2_GPDO371)	8	R/W	00h	16.2.14/ 755
1474	GPIO Pad Data Out Register (SIUL2_GPDO372)	8	R/W	00h	16.2.14/ 755
1475	GPIO Pad Data Out Register (SIUL2_GPDO373)	8	R/W	00h	16.2.14/ 755
1476	GPIO Pad Data Out Register (SIUL2_GPDO374)	8	R/W	00h	16.2.14/ 755
1477	GPIO Pad Data Out Register (SIUL2_GPDO375)	8	R/W	00h	16.2.14/ 755
1478	GPIO Pad Data Out Register (SIUL2_GPDO376)	8	R/W	00h	16.2.14/ 755
1479	GPIO Pad Data Out Register (SIUL2_GPDO377)	8	R/W	00h	16.2.14/ 755
147A	GPIO Pad Data Out Register (SIUL2_GPDO378)	8	R/W	00h	16.2.14/ 755
147B	GPIO Pad Data Out Register (SIUL2_GPDO379)	8	R/W	00h	16.2.14/ 755
147C	GPIO Pad Data Out Register (SIUL2_GPDO380)	8	R/W	00h	16.2.14/ 755
147D	GPIO Pad Data Out Register (SIUL2_GPDO381)	8	R/W	00h	16.2.14/ 755
147E	GPIO Pad Data Out Register (SIUL2_GPDO382)	8	R/W	00h	16.2.14/ 755
147F	GPIO Pad Data Out Register (SIUL2_GPDO383)	8	R/W	00h	16.2.14/ 755
1480	GPIO Pad Data Out Register (SIUL2_GPDO384)	8	R/W	00h	16.2.14/ 755
1481	GPIO Pad Data Out Register (SIUL2_GPDO385)	8	R/W	00h	16.2.14/ 755
1482	GPIO Pad Data Out Register (SIUL2_GPDO386)	8	R/W	00h	16.2.14/ 755
1483	GPIO Pad Data Out Register (SIUL2_GPDO387)	8	R/W	00h	16.2.14/ 755
1484	GPIO Pad Data Out Register (SIUL2_GPDO388)	8	R/W	00h	16.2.14/ 755
1485	GPIO Pad Data Out Register (SIUL2_GPDO389)	8	R/W	00h	16.2.14/ 755
1486	GPIO Pad Data Out Register (SIUL2_GPDO390)	8	R/W	00h	16.2.14/ 755
1487	GPIO Pad Data Out Register (SIUL2_GPDO391)	8	R/W	00h	16.2.14/ 755
1488	GPIO Pad Data Out Register (SIUL2_GPDO392)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1489	GPIO Pad Data Out Register (SIUL2_GPDO393)	8	R/W	00h	16.2.14/ 755
148A	GPIO Pad Data Out Register (SIUL2_GPDO394)	8	R/W	00h	16.2.14/ 755
148B	GPIO Pad Data Out Register (SIUL2_GPDO395)	8	R/W	00h	16.2.14/ 755
148C	GPIO Pad Data Out Register (SIUL2_GPDO396)	8	R/W	00h	16.2.14/ 755
148D	GPIO Pad Data Out Register (SIUL2_GPDO397)	8	R/W	00h	16.2.14/ 755
148E	GPIO Pad Data Out Register (SIUL2_GPDO398)	8	R/W	00h	16.2.14/ 755
148F	GPIO Pad Data Out Register (SIUL2_GPDO399)	8	R/W	00h	16.2.14/ 755
1490	GPIO Pad Data Out Register (SIUL2_GPDO400)	8	R/W	00h	16.2.14/ 755
1491	GPIO Pad Data Out Register (SIUL2_GPDO401)	8	R/W	00h	16.2.14/ 755
1492	GPIO Pad Data Out Register (SIUL2_GPDO402)	8	R/W	00h	16.2.14/ 755
1493	GPIO Pad Data Out Register (SIUL2_GPDO403)	8	R/W	00h	16.2.14/ 755
1494	GPIO Pad Data Out Register (SIUL2_GPDO404)	8	R/W	00h	16.2.14/ 755
1495	GPIO Pad Data Out Register (SIUL2_GPDO405)	8	R/W	00h	16.2.14/ 755
1496	GPIO Pad Data Out Register (SIUL2_GPDO406)	8	R/W	00h	16.2.14/ 755
1497	GPIO Pad Data Out Register (SIUL2_GPDO407)	8	R/W	00h	16.2.14/ 755
1498	GPIO Pad Data Out Register (SIUL2_GPDO408)	8	R/W	00h	16.2.14/ 755
1499	GPIO Pad Data Out Register (SIUL2_GPDO409)	8	R/W	00h	16.2.14/ 755
149A	GPIO Pad Data Out Register (SIUL2_GPDO410)	8	R/W	00h	16.2.14/ 755
149B	GPIO Pad Data Out Register (SIUL2_GPDO411)	8	R/W	00h	16.2.14/ 755
149C	GPIO Pad Data Out Register (SIUL2_GPDO412)	8	R/W	00h	16.2.14/ 755
149D	GPIO Pad Data Out Register (SIUL2_GPDO413)	8	R/W	00h	16.2.14/ 755
149E	GPIO Pad Data Out Register (SIUL2_GPDO414)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
149F	GPIO Pad Data Out Register (SIUL2_GPDO415)	8	R/W	00h	16.2.14/ 755
14A0	GPIO Pad Data Out Register (SIUL2_GPDO416)	8	R/W	00h	16.2.14/ 755
14A1	GPIO Pad Data Out Register (SIUL2_GPDO417)	8	R/W	00h	16.2.14/ 755
14A2	GPIO Pad Data Out Register (SIUL2_GPDO418)	8	R/W	00h	16.2.14/ 755
14A3	GPIO Pad Data Out Register (SIUL2_GPDO419)	8	R/W	00h	16.2.14/ 755
14A4	GPIO Pad Data Out Register (SIUL2_GPDO420)	8	R/W	00h	16.2.14/ 755
14A5	GPIO Pad Data Out Register (SIUL2_GPDO421)	8	R/W	00h	16.2.14/ 755
14A6	GPIO Pad Data Out Register (SIUL2_GPDO422)	8	R/W	00h	16.2.14/ 755
14A7	GPIO Pad Data Out Register (SIUL2_GPDO423)	8	R/W	00h	16.2.14/ 755
14A8	GPIO Pad Data Out Register (SIUL2_GPDO424)	8	R/W	00h	16.2.14/ 755
14A9	GPIO Pad Data Out Register (SIUL2_GPDO425)	8	R/W	00h	16.2.14/ 755
14AA	GPIO Pad Data Out Register (SIUL2_GPDO426)	8	R/W	00h	16.2.14/ 755
14AB	GPIO Pad Data Out Register (SIUL2_GPDO427)	8	R/W	00h	16.2.14/ 755
14AC	GPIO Pad Data Out Register (SIUL2_GPDO428)	8	R/W	00h	16.2.14/ 755
14AD	GPIO Pad Data Out Register (SIUL2_GPDO429)	8	R/W	00h	16.2.14/ 755
14AE	GPIO Pad Data Out Register (SIUL2_GPDO430)	8	R/W	00h	16.2.14/ 755
14AF	GPIO Pad Data Out Register (SIUL2_GPDO431)	8	R/W	00h	16.2.14/ 755
14B0	GPIO Pad Data Out Register (SIUL2_GPDO432)	8	R/W	00h	16.2.14/ 755
14B1	GPIO Pad Data Out Register (SIUL2_GPDO433)	8	R/W	00h	16.2.14/ 755
14B2	GPIO Pad Data Out Register (SIUL2_GPDO434)	8	R/W	00h	16.2.14/ 755
14B3	GPIO Pad Data Out Register (SIUL2_GPDO435)	8	R/W	00h	16.2.14/ 755
14B4	GPIO Pad Data Out Register (SIUL2_GPDO436)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14B5	GPIO Pad Data Out Register (SIUL2_GPDO437)	8	R/W	00h	16.2.14/ 755
14B6	GPIO Pad Data Out Register (SIUL2_GPDO438)	8	R/W	00h	16.2.14/ 755
14B7	GPIO Pad Data Out Register (SIUL2_GPDO439)	8	R/W	00h	16.2.14/ 755
14B8	GPIO Pad Data Out Register (SIUL2_GPDO440)	8	R/W	00h	16.2.14/ 755
14B9	GPIO Pad Data Out Register (SIUL2_GPDO441)	8	R/W	00h	16.2.14/ 755
14BA	GPIO Pad Data Out Register (SIUL2_GPDO442)	8	R/W	00h	16.2.14/ 755
14BB	GPIO Pad Data Out Register (SIUL2_GPDO443)	8	R/W	00h	16.2.14/ 755
14BC	GPIO Pad Data Out Register (SIUL2_GPDO444)	8	R/W	00h	16.2.14/ 755
14BD	GPIO Pad Data Out Register (SIUL2_GPDO445)	8	R/W	00h	16.2.14/ 755
14BE	GPIO Pad Data Out Register (SIUL2_GPDO446)	8	R/W	00h	16.2.14/ 755
14BF	GPIO Pad Data Out Register (SIUL2_GPDO447)	8	R/W	00h	16.2.14/ 755
14C0	GPIO Pad Data Out Register (SIUL2_GPDO448)	8	R/W	00h	16.2.14/ 755
14C1	GPIO Pad Data Out Register (SIUL2_GPDO449)	8	R/W	00h	16.2.14/ 755
14C2	GPIO Pad Data Out Register (SIUL2_GPDO450)	8	R/W	00h	16.2.14/ 755
14C3	GPIO Pad Data Out Register (SIUL2_GPDO451)	8	R/W	00h	16.2.14/ 755
14C4	GPIO Pad Data Out Register (SIUL2_GPDO452)	8	R/W	00h	16.2.14/ 755
14C5	GPIO Pad Data Out Register (SIUL2_GPDO453)	8	R/W	00h	16.2.14/ 755
14C6	GPIO Pad Data Out Register (SIUL2_GPDO454)	8	R/W	00h	16.2.14/ 755
14C7	GPIO Pad Data Out Register (SIUL2_GPDO455)	8	R/W	00h	16.2.14/ 755
14C8	GPIO Pad Data Out Register (SIUL2_GPDO456)	8	R/W	00h	16.2.14/ 755
14C9	GPIO Pad Data Out Register (SIUL2_GPDO457)	8	R/W	00h	16.2.14/ 755
14CA	GPIO Pad Data Out Register (SIUL2_GPDO458)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14CB	GPIO Pad Data Out Register (SIUL2_GPDO459)	8	R/W	00h	16.2.14/ 755
14CC	GPIO Pad Data Out Register (SIUL2_GPDO460)	8	R/W	00h	16.2.14/ 755
14CD	GPIO Pad Data Out Register (SIUL2_GPDO461)	8	R/W	00h	16.2.14/ 755
14CE	GPIO Pad Data Out Register (SIUL2_GPDO462)	8	R/W	00h	16.2.14/ 755
14CF	GPIO Pad Data Out Register (SIUL2_GPDO463)	8	R/W	00h	16.2.14/ 755
14D0	GPIO Pad Data Out Register (SIUL2_GPDO464)	8	R/W	00h	16.2.14/ 755
14D1	GPIO Pad Data Out Register (SIUL2_GPDO465)	8	R/W	00h	16.2.14/ 755
14D2	GPIO Pad Data Out Register (SIUL2_GPDO466)	8	R/W	00h	16.2.14/ 755
14D3	GPIO Pad Data Out Register (SIUL2_GPDO467)	8	R/W	00h	16.2.14/ 755
14D4	GPIO Pad Data Out Register (SIUL2_GPDO468)	8	R/W	00h	16.2.14/ 755
14D5	GPIO Pad Data Out Register (SIUL2_GPDO469)	8	R/W	00h	16.2.14/ 755
14D6	GPIO Pad Data Out Register (SIUL2_GPDO470)	8	R/W	00h	16.2.14/ 755
14D7	GPIO Pad Data Out Register (SIUL2_GPDO471)	8	R/W	00h	16.2.14/ 755
14D8	GPIO Pad Data Out Register (SIUL2_GPDO472)	8	R/W	00h	16.2.14/ 755
14D9	GPIO Pad Data Out Register (SIUL2_GPDO473)	8	R/W	00h	16.2.14/ 755
14DA	GPIO Pad Data Out Register (SIUL2_GPDO474)	8	R/W	00h	16.2.14/ 755
14DB	GPIO Pad Data Out Register (SIUL2_GPDO475)	8	R/W	00h	16.2.14/ 755
14DC	GPIO Pad Data Out Register (SIUL2_GPDO476)	8	R/W	00h	16.2.14/ 755
14DD	GPIO Pad Data Out Register (SIUL2_GPDO477)	8	R/W	00h	16.2.14/ 755
14DE	GPIO Pad Data Out Register (SIUL2_GPDO478)	8	R/W	00h	16.2.14/ 755
14DF	GPIO Pad Data Out Register (SIUL2_GPDO479)	8	R/W	00h	16.2.14/ 755
14E0	GPIO Pad Data Out Register (SIUL2_GPDO480)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14E1	GPIO Pad Data Out Register (SIUL2_GPDO481)	8	R/W	00h	16.2.14/ 755
14E2	GPIO Pad Data Out Register (SIUL2_GPDO482)	8	R/W	00h	16.2.14/ 755
14E3	GPIO Pad Data Out Register (SIUL2_GPDO483)	8	R/W	00h	16.2.14/ 755
14E4	GPIO Pad Data Out Register (SIUL2_GPDO484)	8	R/W	00h	16.2.14/ 755
14E5	GPIO Pad Data Out Register (SIUL2_GPDO485)	8	R/W	00h	16.2.14/ 755
14E6	GPIO Pad Data Out Register (SIUL2_GPDO486)	8	R/W	00h	16.2.14/ 755
14E7	GPIO Pad Data Out Register (SIUL2_GPDO487)	8	R/W	00h	16.2.14/ 755
14E8	GPIO Pad Data Out Register (SIUL2_GPDO488)	8	R/W	00h	16.2.14/ 755
14E9	GPIO Pad Data Out Register (SIUL2_GPDO489)	8	R/W	00h	16.2.14/ 755
14EA	GPIO Pad Data Out Register (SIUL2_GPDO490)	8	R/W	00h	16.2.14/ 755
14EB	GPIO Pad Data Out Register (SIUL2_GPDO491)	8	R/W	00h	16.2.14/ 755
14EC	GPIO Pad Data Out Register (SIUL2_GPDO492)	8	R/W	00h	16.2.14/ 755
14ED	GPIO Pad Data Out Register (SIUL2_GPDO493)	8	R/W	00h	16.2.14/ 755
14EE	GPIO Pad Data Out Register (SIUL2_GPDO494)	8	R/W	00h	16.2.14/ 755
14EF	GPIO Pad Data Out Register (SIUL2_GPDO495)	8	R/W	00h	16.2.14/ 755
14F0	GPIO Pad Data Out Register (SIUL2_GPDO496)	8	R/W	00h	16.2.14/ 755
14F1	GPIO Pad Data Out Register (SIUL2_GPDO497)	8	R/W	00h	16.2.14/ 755
14F2	GPIO Pad Data Out Register (SIUL2_GPDO498)	8	R/W	00h	16.2.14/ 755
14F3	GPIO Pad Data Out Register (SIUL2_GPDO499)	8	R/W	00h	16.2.14/ 755
14F4	GPIO Pad Data Out Register (SIUL2_GPDO500)	8	R/W	00h	16.2.14/ 755
14F5	GPIO Pad Data Out Register (SIUL2_GPDO501)	8	R/W	00h	16.2.14/ 755
14F6	GPIO Pad Data Out Register (SIUL2_GPDO502)	8	R/W	00h	16.2.14/ 755

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14F7	GPIO Pad Data Out Register (SIUL2_GPDO503)	8	R/W	00h	16.2.14/ 755
14F8	GPIO Pad Data Out Register (SIUL2_GPDO504)	8	R/W	00h	16.2.14/ 755
14F9	GPIO Pad Data Out Register (SIUL2_GPDO505)	8	R/W	00h	16.2.14/ 755
14FA	GPIO Pad Data Out Register (SIUL2_GPDO506)	8	R/W	00h	16.2.14/ 755
14FB	GPIO Pad Data Out Register (SIUL2_GPDO507)	8	R/W	00h	16.2.14/ 755
14FC	GPIO Pad Data Out Register (SIUL2_GPDO508)	8	R/W	00h	16.2.14/ 755
14FD	GPIO Pad Data Out Register (SIUL2_GPDO509)	8	R/W	00h	16.2.14/ 755
14FE	GPIO Pad Data Out Register (SIUL2_GPDO510)	8	R/W	00h	16.2.14/ 755
14FF	GPIO Pad Data Out Register (SIUL2_GPDO511)	8	R/W	00h	16.2.14/ 755
1500	GPIO Pad Data In Register (SIUL2_GPDI0)	8	R	00h	16.2.15/ 756
1501	GPIO Pad Data In Register (SIUL2_GPDI1)	8	R	00h	16.2.15/ 756
1502	GPIO Pad Data In Register (SIUL2_GPDI2)	8	R	00h	16.2.15/ 756
1503	GPIO Pad Data In Register (SIUL2_GPDI3)	8	R	00h	16.2.15/ 756
1504	GPIO Pad Data In Register (SIUL2_GPDI4)	8	R	00h	16.2.15/ 756
1505	GPIO Pad Data In Register (SIUL2_GPDI5)	8	R	00h	16.2.15/ 756
1506	GPIO Pad Data In Register (SIUL2_GPDI6)	8	R	00h	16.2.15/ 756
1507	GPIO Pad Data In Register (SIUL2_GPDI7)	8	R	00h	16.2.15/ 756
1508	GPIO Pad Data In Register (SIUL2_GPDI8)	8	R	00h	16.2.15/ 756
1509	GPIO Pad Data In Register (SIUL2_GPDI9)	8	R	00h	16.2.15/ 756
150A	GPIO Pad Data In Register (SIUL2_GPDI10)	8	R	00h	16.2.15/ 756
150B	GPIO Pad Data In Register (SIUL2_GPDI11)	8	R	00h	16.2.15/ 756
150C	GPIO Pad Data In Register (SIUL2_GPDI12)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
150D	GPIO Pad Data In Register (SIUL2_GPDI13)	8	R	00h	16.2.15/ 756
150E	GPIO Pad Data In Register (SIUL2_GPDI14)	8	R	00h	16.2.15/ 756
150F	GPIO Pad Data In Register (SIUL2_GPDI15)	8	R	00h	16.2.15/ 756
1510	GPIO Pad Data In Register (SIUL2_GPDI16)	8	R	00h	16.2.15/ 756
1511	GPIO Pad Data In Register (SIUL2_GPDI17)	8	R	00h	16.2.15/ 756
1512	GPIO Pad Data In Register (SIUL2_GPDI18)	8	R	00h	16.2.15/ 756
1513	GPIO Pad Data In Register (SIUL2_GPDI19)	8	R	00h	16.2.15/ 756
1514	GPIO Pad Data In Register (SIUL2_GPDI20)	8	R	00h	16.2.15/ 756
1515	GPIO Pad Data In Register (SIUL2_GPDI21)	8	R	00h	16.2.15/ 756
1516	GPIO Pad Data In Register (SIUL2_GPDI22)	8	R	00h	16.2.15/ 756
1517	GPIO Pad Data In Register (SIUL2_GPDI23)	8	R	00h	16.2.15/ 756
1518	GPIO Pad Data In Register (SIUL2_GPDI24)	8	R	00h	16.2.15/ 756
1519	GPIO Pad Data In Register (SIUL2_GPDI25)	8	R	00h	16.2.15/ 756
151A	GPIO Pad Data In Register (SIUL2_GPDI26)	8	R	00h	16.2.15/ 756
151B	GPIO Pad Data In Register (SIUL2_GPDI27)	8	R	00h	16.2.15/ 756
151C	GPIO Pad Data In Register (SIUL2_GPDI28)	8	R	00h	16.2.15/ 756
151D	GPIO Pad Data In Register (SIUL2_GPDI29)	8	R	00h	16.2.15/ 756
151E	GPIO Pad Data In Register (SIUL2_GPDI30)	8	R	00h	16.2.15/ 756
151F	GPIO Pad Data In Register (SIUL2_GPDI31)	8	R	00h	16.2.15/ 756
1520	GPIO Pad Data In Register (SIUL2_GPDI32)	8	R	00h	16.2.15/ 756
1521	GPIO Pad Data In Register (SIUL2_GPDI33)	8	R	00h	16.2.15/ 756
1522	GPIO Pad Data In Register (SIUL2_GPDI34)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1523	GPIO Pad Data In Register (SIUL2_GPDI35)	8	R	00h	16.2.15/ 756
1524	GPIO Pad Data In Register (SIUL2_GPDI36)	8	R	00h	16.2.15/ 756
1525	GPIO Pad Data In Register (SIUL2_GPDI37)	8	R	00h	16.2.15/ 756
1526	GPIO Pad Data In Register (SIUL2_GPDI38)	8	R	00h	16.2.15/ 756
1527	GPIO Pad Data In Register (SIUL2_GPDI39)	8	R	00h	16.2.15/ 756
1528	GPIO Pad Data In Register (SIUL2_GPDI40)	8	R	00h	16.2.15/ 756
1529	GPIO Pad Data In Register (SIUL2_GPDI41)	8	R	00h	16.2.15/ 756
152A	GPIO Pad Data In Register (SIUL2_GPDI42)	8	R	00h	16.2.15/ 756
152B	GPIO Pad Data In Register (SIUL2_GPDI43)	8	R	00h	16.2.15/ 756
152C	GPIO Pad Data In Register (SIUL2_GPDI44)	8	R	00h	16.2.15/ 756
152D	GPIO Pad Data In Register (SIUL2_GPDI45)	8	R	00h	16.2.15/ 756
152E	GPIO Pad Data In Register (SIUL2_GPDI46)	8	R	00h	16.2.15/ 756
152F	GPIO Pad Data In Register (SIUL2_GPDI47)	8	R	00h	16.2.15/ 756
1530	GPIO Pad Data In Register (SIUL2_GPDI48)	8	R	00h	16.2.15/ 756
1531	GPIO Pad Data In Register (SIUL2_GPDI49)	8	R	00h	16.2.15/ 756
1532	GPIO Pad Data In Register (SIUL2_GPDI50)	8	R	00h	16.2.15/ 756
1533	GPIO Pad Data In Register (SIUL2_GPDI51)	8	R	00h	16.2.15/ 756
1534	GPIO Pad Data In Register (SIUL2_GPDI52)	8	R	00h	16.2.15/ 756
1535	GPIO Pad Data In Register (SIUL2_GPDI53)	8	R	00h	16.2.15/ 756
1536	GPIO Pad Data In Register (SIUL2_GPDI54)	8	R	00h	16.2.15/ 756
1537	GPIO Pad Data In Register (SIUL2_GPDI55)	8	R	00h	16.2.15/ 756
1538	GPIO Pad Data In Register (SIUL2_GPDI56)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1539	GPIO Pad Data In Register (SIUL2_GPDI57)	8	R	00h	16.2.15/ 756
153A	GPIO Pad Data In Register (SIUL2_GPDI58)	8	R	00h	16.2.15/ 756
153B	GPIO Pad Data In Register (SIUL2_GPDI59)	8	R	00h	16.2.15/ 756
153C	GPIO Pad Data In Register (SIUL2_GPDI60)	8	R	00h	16.2.15/ 756
153D	GPIO Pad Data In Register (SIUL2_GPDI61)	8	R	00h	16.2.15/ 756
153E	GPIO Pad Data In Register (SIUL2_GPDI62)	8	R	00h	16.2.15/ 756
153F	GPIO Pad Data In Register (SIUL2_GPDI63)	8	R	00h	16.2.15/ 756
1540	GPIO Pad Data In Register (SIUL2_GPDI64)	8	R	00h	16.2.15/ 756
1541	GPIO Pad Data In Register (SIUL2_GPDI65)	8	R	00h	16.2.15/ 756
1542	GPIO Pad Data In Register (SIUL2_GPDI66)	8	R	00h	16.2.15/ 756
1543	GPIO Pad Data In Register (SIUL2_GPDI67)	8	R	00h	16.2.15/ 756
1544	GPIO Pad Data In Register (SIUL2_GPDI68)	8	R	00h	16.2.15/ 756
1545	GPIO Pad Data In Register (SIUL2_GPDI69)	8	R	00h	16.2.15/ 756
1546	GPIO Pad Data In Register (SIUL2_GPDI70)	8	R	00h	16.2.15/ 756
1547	GPIO Pad Data In Register (SIUL2_GPDI71)	8	R	00h	16.2.15/ 756
1548	GPIO Pad Data In Register (SIUL2_GPDI72)	8	R	00h	16.2.15/ 756
1549	GPIO Pad Data In Register (SIUL2_GPDI73)	8	R	00h	16.2.15/ 756
154A	GPIO Pad Data In Register (SIUL2_GPDI74)	8	R	00h	16.2.15/ 756
154B	GPIO Pad Data In Register (SIUL2_GPDI75)	8	R	00h	16.2.15/ 756
154C	GPIO Pad Data In Register (SIUL2_GPDI76)	8	R	00h	16.2.15/ 756
154D	GPIO Pad Data In Register (SIUL2_GPDI77)	8	R	00h	16.2.15/ 756
154E	GPIO Pad Data In Register (SIUL2_GPDI78)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
154F	GPIO Pad Data In Register (SIUL2_GPDI79)	8	R	00h	16.2.15/ 756
1550	GPIO Pad Data In Register (SIUL2_GPDI80)	8	R	00h	16.2.15/ 756
1551	GPIO Pad Data In Register (SIUL2_GPDI81)	8	R	00h	16.2.15/ 756
1552	GPIO Pad Data In Register (SIUL2_GPDI82)	8	R	00h	16.2.15/ 756
1553	GPIO Pad Data In Register (SIUL2_GPDI83)	8	R	00h	16.2.15/ 756
1554	GPIO Pad Data In Register (SIUL2_GPDI84)	8	R	00h	16.2.15/ 756
1555	GPIO Pad Data In Register (SIUL2_GPDI85)	8	R	00h	16.2.15/ 756
1556	GPIO Pad Data In Register (SIUL2_GPDI86)	8	R	00h	16.2.15/ 756
1557	GPIO Pad Data In Register (SIUL2_GPDI87)	8	R	00h	16.2.15/ 756
1558	GPIO Pad Data In Register (SIUL2_GPDI88)	8	R	00h	16.2.15/ 756
1559	GPIO Pad Data In Register (SIUL2_GPDI89)	8	R	00h	16.2.15/ 756
155A	GPIO Pad Data In Register (SIUL2_GPDI90)	8	R	00h	16.2.15/ 756
155B	GPIO Pad Data In Register (SIUL2_GPDI91)	8	R	00h	16.2.15/ 756
155C	GPIO Pad Data In Register (SIUL2_GPDI92)	8	R	00h	16.2.15/ 756
155D	GPIO Pad Data In Register (SIUL2_GPDI93)	8	R	00h	16.2.15/ 756
155E	GPIO Pad Data In Register (SIUL2_GPDI94)	8	R	00h	16.2.15/ 756
155F	GPIO Pad Data In Register (SIUL2_GPDI95)	8	R	00h	16.2.15/ 756
1560	GPIO Pad Data In Register (SIUL2_GPDI96)	8	R	00h	16.2.15/ 756
1561	GPIO Pad Data In Register (SIUL2_GPDI97)	8	R	00h	16.2.15/ 756
1562	GPIO Pad Data In Register (SIUL2_GPDI98)	8	R	00h	16.2.15/ 756
1563	GPIO Pad Data In Register (SIUL2_GPDI99)	8	R	00h	16.2.15/ 756
1564	GPIO Pad Data In Register (SIUL2_GPDI100)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1565	GPIO Pad Data In Register (SIUL2_GPDI101)	8	R	00h	16.2.15/ 756
1566	GPIO Pad Data In Register (SIUL2_GPDI102)	8	R	00h	16.2.15/ 756
1567	GPIO Pad Data In Register (SIUL2_GPDI103)	8	R	00h	16.2.15/ 756
1568	GPIO Pad Data In Register (SIUL2_GPDI104)	8	R	00h	16.2.15/ 756
1569	GPIO Pad Data In Register (SIUL2_GPDI105)	8	R	00h	16.2.15/ 756
156A	GPIO Pad Data In Register (SIUL2_GPDI106)	8	R	00h	16.2.15/ 756
156B	GPIO Pad Data In Register (SIUL2_GPDI107)	8	R	00h	16.2.15/ 756
156C	GPIO Pad Data In Register (SIUL2_GPDI108)	8	R	00h	16.2.15/ 756
156D	GPIO Pad Data In Register (SIUL2_GPDI109)	8	R	00h	16.2.15/ 756
156E	GPIO Pad Data In Register (SIUL2_GPDI110)	8	R	00h	16.2.15/ 756
156F	GPIO Pad Data In Register (SIUL2_GPDI111)	8	R	00h	16.2.15/ 756
1570	GPIO Pad Data In Register (SIUL2_GPDI112)	8	R	00h	16.2.15/ 756
1571	GPIO Pad Data In Register (SIUL2_GPDI113)	8	R	00h	16.2.15/ 756
1572	GPIO Pad Data In Register (SIUL2_GPDI114)	8	R	00h	16.2.15/ 756
1573	GPIO Pad Data In Register (SIUL2_GPDI115)	8	R	00h	16.2.15/ 756
1574	GPIO Pad Data In Register (SIUL2_GPDI116)	8	R	00h	16.2.15/ 756
1575	GPIO Pad Data In Register (SIUL2_GPDI117)	8	R	00h	16.2.15/ 756
1576	GPIO Pad Data In Register (SIUL2_GPDI118)	8	R	00h	16.2.15/ 756
1577	GPIO Pad Data In Register (SIUL2_GPDI119)	8	R	00h	16.2.15/ 756
1578	GPIO Pad Data In Register (SIUL2_GPDI120)	8	R	00h	16.2.15/ 756
1579	GPIO Pad Data In Register (SIUL2_GPDI121)	8	R	00h	16.2.15/ 756
157A	GPIO Pad Data In Register (SIUL2_GPDI122)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
157B	GPIO Pad Data In Register (SIUL2_GPDI123)	8	R	00h	16.2.15/ 756
157C	GPIO Pad Data In Register (SIUL2_GPDI124)	8	R	00h	16.2.15/ 756
157D	GPIO Pad Data In Register (SIUL2_GPDI125)	8	R	00h	16.2.15/ 756
157E	GPIO Pad Data In Register (SIUL2_GPDI126)	8	R	00h	16.2.15/ 756
157F	GPIO Pad Data In Register (SIUL2_GPDI127)	8	R	00h	16.2.15/ 756
1580	GPIO Pad Data In Register (SIUL2_GPDI128)	8	R	00h	16.2.15/ 756
1581	GPIO Pad Data In Register (SIUL2_GPDI129)	8	R	00h	16.2.15/ 756
1582	GPIO Pad Data In Register (SIUL2_GPDI130)	8	R	00h	16.2.15/ 756
1583	GPIO Pad Data In Register (SIUL2_GPDI131)	8	R	00h	16.2.15/ 756
1584	GPIO Pad Data In Register (SIUL2_GPDI132)	8	R	00h	16.2.15/ 756
1585	GPIO Pad Data In Register (SIUL2_GPDI133)	8	R	00h	16.2.15/ 756
1586	GPIO Pad Data In Register (SIUL2_GPDI134)	8	R	00h	16.2.15/ 756
1587	GPIO Pad Data In Register (SIUL2_GPDI135)	8	R	00h	16.2.15/ 756
1588	GPIO Pad Data In Register (SIUL2_GPDI136)	8	R	00h	16.2.15/ 756
1589	GPIO Pad Data In Register (SIUL2_GPDI137)	8	R	00h	16.2.15/ 756
158A	GPIO Pad Data In Register (SIUL2_GPDI138)	8	R	00h	16.2.15/ 756
158B	GPIO Pad Data In Register (SIUL2_GPDI139)	8	R	00h	16.2.15/ 756
158C	GPIO Pad Data In Register (SIUL2_GPDI140)	8	R	00h	16.2.15/ 756
158D	GPIO Pad Data In Register (SIUL2_GPDI141)	8	R	00h	16.2.15/ 756
158E	GPIO Pad Data In Register (SIUL2_GPDI142)	8	R	00h	16.2.15/ 756
158F	GPIO Pad Data In Register (SIUL2_GPDI143)	8	R	00h	16.2.15/ 756
1590	GPIO Pad Data In Register (SIUL2_GPDI144)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1591	GPIO Pad Data In Register (SIUL2_GPDI145)	8	R	00h	16.2.15/ 756
1592	GPIO Pad Data In Register (SIUL2_GPDI146)	8	R	00h	16.2.15/ 756
1593	GPIO Pad Data In Register (SIUL2_GPDI147)	8	R	00h	16.2.15/ 756
1594	GPIO Pad Data In Register (SIUL2_GPDI148)	8	R	00h	16.2.15/ 756
1595	GPIO Pad Data In Register (SIUL2_GPDI149)	8	R	00h	16.2.15/ 756
1596	GPIO Pad Data In Register (SIUL2_GPDI150)	8	R	00h	16.2.15/ 756
1597	GPIO Pad Data In Register (SIUL2_GPDI151)	8	R	00h	16.2.15/ 756
1598	GPIO Pad Data In Register (SIUL2_GPDI152)	8	R	00h	16.2.15/ 756
1599	GPIO Pad Data In Register (SIUL2_GPDI153)	8	R	00h	16.2.15/ 756
159A	GPIO Pad Data In Register (SIUL2_GPDI154)	8	R	00h	16.2.15/ 756
159B	GPIO Pad Data In Register (SIUL2_GPDI155)	8	R	00h	16.2.15/ 756
159C	GPIO Pad Data In Register (SIUL2_GPDI156)	8	R	00h	16.2.15/ 756
159D	GPIO Pad Data In Register (SIUL2_GPDI157)	8	R	00h	16.2.15/ 756
159E	GPIO Pad Data In Register (SIUL2_GPDI158)	8	R	00h	16.2.15/ 756
159F	GPIO Pad Data In Register (SIUL2_GPDI159)	8	R	00h	16.2.15/ 756
15A0	GPIO Pad Data In Register (SIUL2_GPDI160)	8	R	00h	16.2.15/ 756
15A1	GPIO Pad Data In Register (SIUL2_GPDI161)	8	R	00h	16.2.15/ 756
15A2	GPIO Pad Data In Register (SIUL2_GPDI162)	8	R	00h	16.2.15/ 756
15A3	GPIO Pad Data In Register (SIUL2_GPDI163)	8	R	00h	16.2.15/ 756
15A4	GPIO Pad Data In Register (SIUL2_GPDI164)	8	R	00h	16.2.15/ 756
15A5	GPIO Pad Data In Register (SIUL2_GPDI165)	8	R	00h	16.2.15/ 756
15A6	GPIO Pad Data In Register (SIUL2_GPDI166)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15A7	GPIO Pad Data In Register (SIUL2_GPDI167)	8	R	00h	16.2.15/ 756
15A8	GPIO Pad Data In Register (SIUL2_GPDI168)	8	R	00h	16.2.15/ 756
15A9	GPIO Pad Data In Register (SIUL2_GPDI169)	8	R	00h	16.2.15/ 756
15AA	GPIO Pad Data In Register (SIUL2_GPDI170)	8	R	00h	16.2.15/ 756
15AB	GPIO Pad Data In Register (SIUL2_GPDI171)	8	R	00h	16.2.15/ 756
15AC	GPIO Pad Data In Register (SIUL2_GPDI172)	8	R	00h	16.2.15/ 756
15AD	GPIO Pad Data In Register (SIUL2_GPDI173)	8	R	00h	16.2.15/ 756
15AE	GPIO Pad Data In Register (SIUL2_GPDI174)	8	R	00h	16.2.15/ 756
15AF	GPIO Pad Data In Register (SIUL2_GPDI175)	8	R	00h	16.2.15/ 756
15B0	GPIO Pad Data In Register (SIUL2_GPDI176)	8	R	00h	16.2.15/ 756
15B1	GPIO Pad Data In Register (SIUL2_GPDI177)	8	R	00h	16.2.15/ 756
15B2	GPIO Pad Data In Register (SIUL2_GPDI178)	8	R	00h	16.2.15/ 756
15B3	GPIO Pad Data In Register (SIUL2_GPDI179)	8	R	00h	16.2.15/ 756
15B4	GPIO Pad Data In Register (SIUL2_GPDI180)	8	R	00h	16.2.15/ 756
15B5	GPIO Pad Data In Register (SIUL2_GPDI181)	8	R	00h	16.2.15/ 756
15B6	GPIO Pad Data In Register (SIUL2_GPDI182)	8	R	00h	16.2.15/ 756
15B7	GPIO Pad Data In Register (SIUL2_GPDI183)	8	R	00h	16.2.15/ 756
15B8	GPIO Pad Data In Register (SIUL2_GPDI184)	8	R	00h	16.2.15/ 756
15B9	GPIO Pad Data In Register (SIUL2_GPDI185)	8	R	00h	16.2.15/ 756
15BA	GPIO Pad Data In Register (SIUL2_GPDI186)	8	R	00h	16.2.15/ 756
15BB	GPIO Pad Data In Register (SIUL2_GPDI187)	8	R	00h	16.2.15/ 756
15BC	GPIO Pad Data In Register (SIUL2_GPDI188)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15BD	GPIO Pad Data In Register (SIUL2_GPDI189)	8	R	00h	16.2.15/ 756
15BE	GPIO Pad Data In Register (SIUL2_GPDI190)	8	R	00h	16.2.15/ 756
15BF	GPIO Pad Data In Register (SIUL2_GPDI191)	8	R	00h	16.2.15/ 756
15C0	GPIO Pad Data In Register (SIUL2_GPDI192)	8	R	00h	16.2.15/ 756
15C1	GPIO Pad Data In Register (SIUL2_GPDI193)	8	R	00h	16.2.15/ 756
15C2	GPIO Pad Data In Register (SIUL2_GPDI194)	8	R	00h	16.2.15/ 756
15C3	GPIO Pad Data In Register (SIUL2_GPDI195)	8	R	00h	16.2.15/ 756
15C4	GPIO Pad Data In Register (SIUL2_GPDI196)	8	R	00h	16.2.15/ 756
15C5	GPIO Pad Data In Register (SIUL2_GPDI197)	8	R	00h	16.2.15/ 756
15C6	GPIO Pad Data In Register (SIUL2_GPDI198)	8	R	00h	16.2.15/ 756
15C7	GPIO Pad Data In Register (SIUL2_GPDI199)	8	R	00h	16.2.15/ 756
15C8	GPIO Pad Data In Register (SIUL2_GPDI200)	8	R	00h	16.2.15/ 756
15C9	GPIO Pad Data In Register (SIUL2_GPDI201)	8	R	00h	16.2.15/ 756
15CA	GPIO Pad Data In Register (SIUL2_GPDI202)	8	R	00h	16.2.15/ 756
15CB	GPIO Pad Data In Register (SIUL2_GPDI203)	8	R	00h	16.2.15/ 756
15CC	GPIO Pad Data In Register (SIUL2_GPDI204)	8	R	00h	16.2.15/ 756
15CD	GPIO Pad Data In Register (SIUL2_GPDI205)	8	R	00h	16.2.15/ 756
15CE	GPIO Pad Data In Register (SIUL2_GPDI206)	8	R	00h	16.2.15/ 756
15CF	GPIO Pad Data In Register (SIUL2_GPDI207)	8	R	00h	16.2.15/ 756
15D0	GPIO Pad Data In Register (SIUL2_GPDI208)	8	R	00h	16.2.15/ 756
15D1	GPIO Pad Data In Register (SIUL2_GPDI209)	8	R	00h	16.2.15/ 756
15D2	GPIO Pad Data In Register (SIUL2_GPDI210)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15D3	GPIO Pad Data In Register (SIUL2_GPDI211)	8	R	00h	16.2.15/ 756
15D4	GPIO Pad Data In Register (SIUL2_GPDI212)	8	R	00h	16.2.15/ 756
15D5	GPIO Pad Data In Register (SIUL2_GPDI213)	8	R	00h	16.2.15/ 756
15D6	GPIO Pad Data In Register (SIUL2_GPDI214)	8	R	00h	16.2.15/ 756
15D7	GPIO Pad Data In Register (SIUL2_GPDI215)	8	R	00h	16.2.15/ 756
15D8	GPIO Pad Data In Register (SIUL2_GPDI216)	8	R	00h	16.2.15/ 756
15D9	GPIO Pad Data In Register (SIUL2_GPDI217)	8	R	00h	16.2.15/ 756
15DA	GPIO Pad Data In Register (SIUL2_GPDI218)	8	R	00h	16.2.15/ 756
15DB	GPIO Pad Data In Register (SIUL2_GPDI219)	8	R	00h	16.2.15/ 756
15DC	GPIO Pad Data In Register (SIUL2_GPDI220)	8	R	00h	16.2.15/ 756
15DD	GPIO Pad Data In Register (SIUL2_GPDI221)	8	R	00h	16.2.15/ 756
15DE	GPIO Pad Data In Register (SIUL2_GPDI222)	8	R	00h	16.2.15/ 756
15DF	GPIO Pad Data In Register (SIUL2_GPDI223)	8	R	00h	16.2.15/ 756
15E0	GPIO Pad Data In Register (SIUL2_GPDI224)	8	R	00h	16.2.15/ 756
15E1	GPIO Pad Data In Register (SIUL2_GPDI225)	8	R	00h	16.2.15/ 756
15E2	GPIO Pad Data In Register (SIUL2_GPDI226)	8	R	00h	16.2.15/ 756
15E3	GPIO Pad Data In Register (SIUL2_GPDI227)	8	R	00h	16.2.15/ 756
15E4	GPIO Pad Data In Register (SIUL2_GPDI228)	8	R	00h	16.2.15/ 756
15E5	GPIO Pad Data In Register (SIUL2_GPDI229)	8	R	00h	16.2.15/ 756
15E6	GPIO Pad Data In Register (SIUL2_GPDI230)	8	R	00h	16.2.15/ 756
15E7	GPIO Pad Data In Register (SIUL2_GPDI231)	8	R	00h	16.2.15/ 756
15E8	GPIO Pad Data In Register (SIUL2_GPDI232)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15E9	GPIO Pad Data In Register (SIUL2_GPDI233)	8	R	00h	16.2.15/ 756
15EA	GPIO Pad Data In Register (SIUL2_GPDI234)	8	R	00h	16.2.15/ 756
15EB	GPIO Pad Data In Register (SIUL2_GPDI235)	8	R	00h	16.2.15/ 756
15EC	GPIO Pad Data In Register (SIUL2_GPDI236)	8	R	00h	16.2.15/ 756
15ED	GPIO Pad Data In Register (SIUL2_GPDI237)	8	R	00h	16.2.15/ 756
15EE	GPIO Pad Data In Register (SIUL2_GPDI238)	8	R	00h	16.2.15/ 756
15EF	GPIO Pad Data In Register (SIUL2_GPDI239)	8	R	00h	16.2.15/ 756
15F0	GPIO Pad Data In Register (SIUL2_GPDI240)	8	R	00h	16.2.15/ 756
15F1	GPIO Pad Data In Register (SIUL2_GPDI241)	8	R	00h	16.2.15/ 756
15F2	GPIO Pad Data In Register (SIUL2_GPDI242)	8	R	00h	16.2.15/ 756
15F3	GPIO Pad Data In Register (SIUL2_GPDI243)	8	R	00h	16.2.15/ 756
15F4	GPIO Pad Data In Register (SIUL2_GPDI244)	8	R	00h	16.2.15/ 756
15F5	GPIO Pad Data In Register (SIUL2_GPDI245)	8	R	00h	16.2.15/ 756
15F6	GPIO Pad Data In Register (SIUL2_GPDI246)	8	R	00h	16.2.15/ 756
15F7	GPIO Pad Data In Register (SIUL2_GPDI247)	8	R	00h	16.2.15/ 756
15F8	GPIO Pad Data In Register (SIUL2_GPDI248)	8	R	00h	16.2.15/ 756
15F9	GPIO Pad Data In Register (SIUL2_GPDI249)	8	R	00h	16.2.15/ 756
15FA	GPIO Pad Data In Register (SIUL2_GPDI250)	8	R	00h	16.2.15/ 756
15FB	GPIO Pad Data In Register (SIUL2_GPDI251)	8	R	00h	16.2.15/ 756
15FC	GPIO Pad Data In Register (SIUL2_GPDI252)	8	R	00h	16.2.15/ 756
15FD	GPIO Pad Data In Register (SIUL2_GPDI253)	8	R	00h	16.2.15/ 756
15FE	GPIO Pad Data In Register (SIUL2_GPDI254)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15FF	GPIO Pad Data In Register (SIUL2_GPDI255)	8	R	00h	16.2.15/ 756
1600	GPIO Pad Data In Register (SIUL2_GPDI256)	8	R	00h	16.2.15/ 756
1601	GPIO Pad Data In Register (SIUL2_GPDI257)	8	R	00h	16.2.15/ 756
1602	GPIO Pad Data In Register (SIUL2_GPDI258)	8	R	00h	16.2.15/ 756
1603	GPIO Pad Data In Register (SIUL2_GPDI259)	8	R	00h	16.2.15/ 756
1604	GPIO Pad Data In Register (SIUL2_GPDI260)	8	R	00h	16.2.15/ 756
1605	GPIO Pad Data In Register (SIUL2_GPDI261)	8	R	00h	16.2.15/ 756
1606	GPIO Pad Data In Register (SIUL2_GPDI262)	8	R	00h	16.2.15/ 756
1607	GPIO Pad Data In Register (SIUL2_GPDI263)	8	R	00h	16.2.15/ 756
1608	GPIO Pad Data In Register (SIUL2_GPDI264)	8	R	00h	16.2.15/ 756
1609	GPIO Pad Data In Register (SIUL2_GPDI265)	8	R	00h	16.2.15/ 756
160A	GPIO Pad Data In Register (SIUL2_GPDI266)	8	R	00h	16.2.15/ 756
160B	GPIO Pad Data In Register (SIUL2_GPDI267)	8	R	00h	16.2.15/ 756
160C	GPIO Pad Data In Register (SIUL2_GPDI268)	8	R	00h	16.2.15/ 756
160D	GPIO Pad Data In Register (SIUL2_GPDI269)	8	R	00h	16.2.15/ 756
160E	GPIO Pad Data In Register (SIUL2_GPDI270)	8	R	00h	16.2.15/ 756
160F	GPIO Pad Data In Register (SIUL2_GPDI271)	8	R	00h	16.2.15/ 756
1610	GPIO Pad Data In Register (SIUL2_GPDI272)	8	R	00h	16.2.15/ 756
1611	GPIO Pad Data In Register (SIUL2_GPDI273)	8	R	00h	16.2.15/ 756
1612	GPIO Pad Data In Register (SIUL2_GPDI274)	8	R	00h	16.2.15/ 756
1613	GPIO Pad Data In Register (SIUL2_GPDI275)	8	R	00h	16.2.15/ 756
1614	GPIO Pad Data In Register (SIUL2_GPDI276)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1615	GPIO Pad Data In Register (SIUL2_GPDI277)	8	R	00h	16.2.15/ 756
1616	GPIO Pad Data In Register (SIUL2_GPDI278)	8	R	00h	16.2.15/ 756
1617	GPIO Pad Data In Register (SIUL2_GPDI279)	8	R	00h	16.2.15/ 756
1618	GPIO Pad Data In Register (SIUL2_GPDI280)	8	R	00h	16.2.15/ 756
1619	GPIO Pad Data In Register (SIUL2_GPDI281)	8	R	00h	16.2.15/ 756
161A	GPIO Pad Data In Register (SIUL2_GPDI282)	8	R	00h	16.2.15/ 756
161B	GPIO Pad Data In Register (SIUL2_GPDI283)	8	R	00h	16.2.15/ 756
161C	GPIO Pad Data In Register (SIUL2_GPDI284)	8	R	00h	16.2.15/ 756
161D	GPIO Pad Data In Register (SIUL2_GPDI285)	8	R	00h	16.2.15/ 756
161E	GPIO Pad Data In Register (SIUL2_GPDI286)	8	R	00h	16.2.15/ 756
161F	GPIO Pad Data In Register (SIUL2_GPDI287)	8	R	00h	16.2.15/ 756
1620	GPIO Pad Data In Register (SIUL2_GPDI288)	8	R	00h	16.2.15/ 756
1621	GPIO Pad Data In Register (SIUL2_GPDI289)	8	R	00h	16.2.15/ 756
1622	GPIO Pad Data In Register (SIUL2_GPDI290)	8	R	00h	16.2.15/ 756
1623	GPIO Pad Data In Register (SIUL2_GPDI291)	8	R	00h	16.2.15/ 756
1624	GPIO Pad Data In Register (SIUL2_GPDI292)	8	R	00h	16.2.15/ 756
1625	GPIO Pad Data In Register (SIUL2_GPDI293)	8	R	00h	16.2.15/ 756
1626	GPIO Pad Data In Register (SIUL2_GPDI294)	8	R	00h	16.2.15/ 756
1627	GPIO Pad Data In Register (SIUL2_GPDI295)	8	R	00h	16.2.15/ 756
1628	GPIO Pad Data In Register (SIUL2_GPDI296)	8	R	00h	16.2.15/ 756
1629	GPIO Pad Data In Register (SIUL2_GPDI297)	8	R	00h	16.2.15/ 756
162A	GPIO Pad Data In Register (SIUL2_GPDI298)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
162B	GPIO Pad Data In Register (SIUL2_GPDI299)	8	R	00h	16.2.15/ 756
162C	GPIO Pad Data In Register (SIUL2_GPDI300)	8	R	00h	16.2.15/ 756
162D	GPIO Pad Data In Register (SIUL2_GPDI301)	8	R	00h	16.2.15/ 756
162E	GPIO Pad Data In Register (SIUL2_GPDI302)	8	R	00h	16.2.15/ 756
162F	GPIO Pad Data In Register (SIUL2_GPDI303)	8	R	00h	16.2.15/ 756
1630	GPIO Pad Data In Register (SIUL2_GPDI304)	8	R	00h	16.2.15/ 756
1631	GPIO Pad Data In Register (SIUL2_GPDI305)	8	R	00h	16.2.15/ 756
1632	GPIO Pad Data In Register (SIUL2_GPDI306)	8	R	00h	16.2.15/ 756
1633	GPIO Pad Data In Register (SIUL2_GPDI307)	8	R	00h	16.2.15/ 756
1634	GPIO Pad Data In Register (SIUL2_GPDI308)	8	R	00h	16.2.15/ 756
1635	GPIO Pad Data In Register (SIUL2_GPDI309)	8	R	00h	16.2.15/ 756
1636	GPIO Pad Data In Register (SIUL2_GPDI310)	8	R	00h	16.2.15/ 756
1637	GPIO Pad Data In Register (SIUL2_GPDI311)	8	R	00h	16.2.15/ 756
1638	GPIO Pad Data In Register (SIUL2_GPDI312)	8	R	00h	16.2.15/ 756
1639	GPIO Pad Data In Register (SIUL2_GPDI313)	8	R	00h	16.2.15/ 756
163A	GPIO Pad Data In Register (SIUL2_GPDI314)	8	R	00h	16.2.15/ 756
163B	GPIO Pad Data In Register (SIUL2_GPDI315)	8	R	00h	16.2.15/ 756
163C	GPIO Pad Data In Register (SIUL2_GPDI316)	8	R	00h	16.2.15/ 756
163D	GPIO Pad Data In Register (SIUL2_GPDI317)	8	R	00h	16.2.15/ 756
163E	GPIO Pad Data In Register (SIUL2_GPDI318)	8	R	00h	16.2.15/ 756
163F	GPIO Pad Data In Register (SIUL2_GPDI319)	8	R	00h	16.2.15/ 756
1640	GPIO Pad Data In Register (SIUL2_GPDI320)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1641	GPIO Pad Data In Register (SIUL2_GPDI321)	8	R	00h	16.2.15/ 756
1642	GPIO Pad Data In Register (SIUL2_GPDI322)	8	R	00h	16.2.15/ 756
1643	GPIO Pad Data In Register (SIUL2_GPDI323)	8	R	00h	16.2.15/ 756
1644	GPIO Pad Data In Register (SIUL2_GPDI324)	8	R	00h	16.2.15/ 756
1645	GPIO Pad Data In Register (SIUL2_GPDI325)	8	R	00h	16.2.15/ 756
1646	GPIO Pad Data In Register (SIUL2_GPDI326)	8	R	00h	16.2.15/ 756
1647	GPIO Pad Data In Register (SIUL2_GPDI327)	8	R	00h	16.2.15/ 756
1648	GPIO Pad Data In Register (SIUL2_GPDI328)	8	R	00h	16.2.15/ 756
1649	GPIO Pad Data In Register (SIUL2_GPDI329)	8	R	00h	16.2.15/ 756
164A	GPIO Pad Data In Register (SIUL2_GPDI330)	8	R	00h	16.2.15/ 756
164B	GPIO Pad Data In Register (SIUL2_GPDI331)	8	R	00h	16.2.15/ 756
164C	GPIO Pad Data In Register (SIUL2_GPDI332)	8	R	00h	16.2.15/ 756
164D	GPIO Pad Data In Register (SIUL2_GPDI333)	8	R	00h	16.2.15/ 756
164E	GPIO Pad Data In Register (SIUL2_GPDI334)	8	R	00h	16.2.15/ 756
164F	GPIO Pad Data In Register (SIUL2_GPDI335)	8	R	00h	16.2.15/ 756
1650	GPIO Pad Data In Register (SIUL2_GPDI336)	8	R	00h	16.2.15/ 756
1651	GPIO Pad Data In Register (SIUL2_GPDI337)	8	R	00h	16.2.15/ 756
1652	GPIO Pad Data In Register (SIUL2_GPDI338)	8	R	00h	16.2.15/ 756
1653	GPIO Pad Data In Register (SIUL2_GPDI339)	8	R	00h	16.2.15/ 756
1654	GPIO Pad Data In Register (SIUL2_GPDI340)	8	R	00h	16.2.15/ 756
1655	GPIO Pad Data In Register (SIUL2_GPDI341)	8	R	00h	16.2.15/ 756
1656	GPIO Pad Data In Register (SIUL2_GPDI342)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1657	GPIO Pad Data In Register (SIUL2_GPDI343)	8	R	00h	16.2.15/ 756
1658	GPIO Pad Data In Register (SIUL2_GPDI344)	8	R	00h	16.2.15/ 756
1659	GPIO Pad Data In Register (SIUL2_GPDI345)	8	R	00h	16.2.15/ 756
165A	GPIO Pad Data In Register (SIUL2_GPDI346)	8	R	00h	16.2.15/ 756
165B	GPIO Pad Data In Register (SIUL2_GPDI347)	8	R	00h	16.2.15/ 756
165C	GPIO Pad Data In Register (SIUL2_GPDI348)	8	R	00h	16.2.15/ 756
165D	GPIO Pad Data In Register (SIUL2_GPDI349)	8	R	00h	16.2.15/ 756
165E	GPIO Pad Data In Register (SIUL2_GPDI350)	8	R	00h	16.2.15/ 756
165F	GPIO Pad Data In Register (SIUL2_GPDI351)	8	R	00h	16.2.15/ 756
1660	GPIO Pad Data In Register (SIUL2_GPDI352)	8	R	00h	16.2.15/ 756
1661	GPIO Pad Data In Register (SIUL2_GPDI353)	8	R	00h	16.2.15/ 756
1662	GPIO Pad Data In Register (SIUL2_GPDI354)	8	R	00h	16.2.15/ 756
1663	GPIO Pad Data In Register (SIUL2_GPDI355)	8	R	00h	16.2.15/ 756
1664	GPIO Pad Data In Register (SIUL2_GPDI356)	8	R	00h	16.2.15/ 756
1665	GPIO Pad Data In Register (SIUL2_GPDI357)	8	R	00h	16.2.15/ 756
1666	GPIO Pad Data In Register (SIUL2_GPDI358)	8	R	00h	16.2.15/ 756
1667	GPIO Pad Data In Register (SIUL2_GPDI359)	8	R	00h	16.2.15/ 756
1668	GPIO Pad Data In Register (SIUL2_GPDI360)	8	R	00h	16.2.15/ 756
1669	GPIO Pad Data In Register (SIUL2_GPDI361)	8	R	00h	16.2.15/ 756
166A	GPIO Pad Data In Register (SIUL2_GPDI362)	8	R	00h	16.2.15/ 756
166B	GPIO Pad Data In Register (SIUL2_GPDI363)	8	R	00h	16.2.15/ 756
166C	GPIO Pad Data In Register (SIUL2_GPDI364)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
166D	GPIO Pad Data In Register (SIUL2_GPDI365)	8	R	00h	16.2.15/ 756
166E	GPIO Pad Data In Register (SIUL2_GPDI366)	8	R	00h	16.2.15/ 756
166F	GPIO Pad Data In Register (SIUL2_GPDI367)	8	R	00h	16.2.15/ 756
1670	GPIO Pad Data In Register (SIUL2_GPDI368)	8	R	00h	16.2.15/ 756
1671	GPIO Pad Data In Register (SIUL2_GPDI369)	8	R	00h	16.2.15/ 756
1672	GPIO Pad Data In Register (SIUL2_GPDI370)	8	R	00h	16.2.15/ 756
1673	GPIO Pad Data In Register (SIUL2_GPDI371)	8	R	00h	16.2.15/ 756
1674	GPIO Pad Data In Register (SIUL2_GPDI372)	8	R	00h	16.2.15/ 756
1675	GPIO Pad Data In Register (SIUL2_GPDI373)	8	R	00h	16.2.15/ 756
1676	GPIO Pad Data In Register (SIUL2_GPDI374)	8	R	00h	16.2.15/ 756
1677	GPIO Pad Data In Register (SIUL2_GPDI375)	8	R	00h	16.2.15/ 756
1678	GPIO Pad Data In Register (SIUL2_GPDI376)	8	R	00h	16.2.15/ 756
1679	GPIO Pad Data In Register (SIUL2_GPDI377)	8	R	00h	16.2.15/ 756
167A	GPIO Pad Data In Register (SIUL2_GPDI378)	8	R	00h	16.2.15/ 756
167B	GPIO Pad Data In Register (SIUL2_GPDI379)	8	R	00h	16.2.15/ 756
167C	GPIO Pad Data In Register (SIUL2_GPDI380)	8	R	00h	16.2.15/ 756
167D	GPIO Pad Data In Register (SIUL2_GPDI381)	8	R	00h	16.2.15/ 756
167E	GPIO Pad Data In Register (SIUL2_GPDI382)	8	R	00h	16.2.15/ 756
167F	GPIO Pad Data In Register (SIUL2_GPDI383)	8	R	00h	16.2.15/ 756
1680	GPIO Pad Data In Register (SIUL2_GPDI384)	8	R	00h	16.2.15/ 756
1681	GPIO Pad Data In Register (SIUL2_GPDI385)	8	R	00h	16.2.15/ 756
1682	GPIO Pad Data In Register (SIUL2_GPDI386)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1683	GPIO Pad Data In Register (SIUL2_GPDI387)	8	R	00h	16.2.15/ 756
1684	GPIO Pad Data In Register (SIUL2_GPDI388)	8	R	00h	16.2.15/ 756
1685	GPIO Pad Data In Register (SIUL2_GPDI389)	8	R	00h	16.2.15/ 756
1686	GPIO Pad Data In Register (SIUL2_GPDI390)	8	R	00h	16.2.15/ 756
1687	GPIO Pad Data In Register (SIUL2_GPDI391)	8	R	00h	16.2.15/ 756
1688	GPIO Pad Data In Register (SIUL2_GPDI392)	8	R	00h	16.2.15/ 756
1689	GPIO Pad Data In Register (SIUL2_GPDI393)	8	R	00h	16.2.15/ 756
168A	GPIO Pad Data In Register (SIUL2_GPDI394)	8	R	00h	16.2.15/ 756
168B	GPIO Pad Data In Register (SIUL2_GPDI395)	8	R	00h	16.2.15/ 756
168C	GPIO Pad Data In Register (SIUL2_GPDI396)	8	R	00h	16.2.15/ 756
168D	GPIO Pad Data In Register (SIUL2_GPDI397)	8	R	00h	16.2.15/ 756
168E	GPIO Pad Data In Register (SIUL2_GPDI398)	8	R	00h	16.2.15/ 756
168F	GPIO Pad Data In Register (SIUL2_GPDI399)	8	R	00h	16.2.15/ 756
1690	GPIO Pad Data In Register (SIUL2_GPDI400)	8	R	00h	16.2.15/ 756
1691	GPIO Pad Data In Register (SIUL2_GPDI401)	8	R	00h	16.2.15/ 756
1692	GPIO Pad Data In Register (SIUL2_GPDI402)	8	R	00h	16.2.15/ 756
1693	GPIO Pad Data In Register (SIUL2_GPDI403)	8	R	00h	16.2.15/ 756
1694	GPIO Pad Data In Register (SIUL2_GPDI404)	8	R	00h	16.2.15/ 756
1695	GPIO Pad Data In Register (SIUL2_GPDI405)	8	R	00h	16.2.15/ 756
1696	GPIO Pad Data In Register (SIUL2_GPDI406)	8	R	00h	16.2.15/ 756
1697	GPIO Pad Data In Register (SIUL2_GPDI407)	8	R	00h	16.2.15/ 756
1698	GPIO Pad Data In Register (SIUL2_GPDI408)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1699	GPIO Pad Data In Register (SIUL2_GPDI409)	8	R	00h	16.2.15/ 756
169A	GPIO Pad Data In Register (SIUL2_GPDI410)	8	R	00h	16.2.15/ 756
169B	GPIO Pad Data In Register (SIUL2_GPDI411)	8	R	00h	16.2.15/ 756
169C	GPIO Pad Data In Register (SIUL2_GPDI412)	8	R	00h	16.2.15/ 756
169D	GPIO Pad Data In Register (SIUL2_GPDI413)	8	R	00h	16.2.15/ 756
169E	GPIO Pad Data In Register (SIUL2_GPDI414)	8	R	00h	16.2.15/ 756
169F	GPIO Pad Data In Register (SIUL2_GPDI415)	8	R	00h	16.2.15/ 756
16A0	GPIO Pad Data In Register (SIUL2_GPDI416)	8	R	00h	16.2.15/ 756
16A1	GPIO Pad Data In Register (SIUL2_GPDI417)	8	R	00h	16.2.15/ 756
16A2	GPIO Pad Data In Register (SIUL2_GPDI418)	8	R	00h	16.2.15/ 756
16A3	GPIO Pad Data In Register (SIUL2_GPDI419)	8	R	00h	16.2.15/ 756
16A4	GPIO Pad Data In Register (SIUL2_GPDI420)	8	R	00h	16.2.15/ 756
16A5	GPIO Pad Data In Register (SIUL2_GPDI421)	8	R	00h	16.2.15/ 756
16A6	GPIO Pad Data In Register (SIUL2_GPDI422)	8	R	00h	16.2.15/ 756
16A7	GPIO Pad Data In Register (SIUL2_GPDI423)	8	R	00h	16.2.15/ 756
16A8	GPIO Pad Data In Register (SIUL2_GPDI424)	8	R	00h	16.2.15/ 756
16A9	GPIO Pad Data In Register (SIUL2_GPDI425)	8	R	00h	16.2.15/ 756
16AA	GPIO Pad Data In Register (SIUL2_GPDI426)	8	R	00h	16.2.15/ 756
16AB	GPIO Pad Data In Register (SIUL2_GPDI427)	8	R	00h	16.2.15/ 756
16AC	GPIO Pad Data In Register (SIUL2_GPDI428)	8	R	00h	16.2.15/ 756
16AD	GPIO Pad Data In Register (SIUL2_GPDI429)	8	R	00h	16.2.15/ 756
16AE	GPIO Pad Data In Register (SIUL2_GPDI430)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16AF	GPIO Pad Data In Register (SIUL2_GPDI431)	8	R	00h	16.2.15/ 756
16B0	GPIO Pad Data In Register (SIUL2_GPDI432)	8	R	00h	16.2.15/ 756
16B1	GPIO Pad Data In Register (SIUL2_GPDI433)	8	R	00h	16.2.15/ 756
16B2	GPIO Pad Data In Register (SIUL2_GPDI434)	8	R	00h	16.2.15/ 756
16B3	GPIO Pad Data In Register (SIUL2_GPDI435)	8	R	00h	16.2.15/ 756
16B4	GPIO Pad Data In Register (SIUL2_GPDI436)	8	R	00h	16.2.15/ 756
16B5	GPIO Pad Data In Register (SIUL2_GPDI437)	8	R	00h	16.2.15/ 756
16B6	GPIO Pad Data In Register (SIUL2_GPDI438)	8	R	00h	16.2.15/ 756
16B7	GPIO Pad Data In Register (SIUL2_GPDI439)	8	R	00h	16.2.15/ 756
16B8	GPIO Pad Data In Register (SIUL2_GPDI440)	8	R	00h	16.2.15/ 756
16B9	GPIO Pad Data In Register (SIUL2_GPDI441)	8	R	00h	16.2.15/ 756
16BA	GPIO Pad Data In Register (SIUL2_GPDI442)	8	R	00h	16.2.15/ 756
16BB	GPIO Pad Data In Register (SIUL2_GPDI443)	8	R	00h	16.2.15/ 756
16BC	GPIO Pad Data In Register (SIUL2_GPDI444)	8	R	00h	16.2.15/ 756
16BD	GPIO Pad Data In Register (SIUL2_GPDI445)	8	R	00h	16.2.15/ 756
16BE	GPIO Pad Data In Register (SIUL2_GPDI446)	8	R	00h	16.2.15/ 756
16BF	GPIO Pad Data In Register (SIUL2_GPDI447)	8	R	00h	16.2.15/ 756
16C0	GPIO Pad Data In Register (SIUL2_GPDI448)	8	R	00h	16.2.15/ 756
16C1	GPIO Pad Data In Register (SIUL2_GPDI449)	8	R	00h	16.2.15/ 756
16C2	GPIO Pad Data In Register (SIUL2_GPDI450)	8	R	00h	16.2.15/ 756
16C3	GPIO Pad Data In Register (SIUL2_GPDI451)	8	R	00h	16.2.15/ 756
16C4	GPIO Pad Data In Register (SIUL2_GPDI452)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16C5	GPIO Pad Data In Register (SIUL2_GPDI453)	8	R	00h	16.2.15/ 756
16C6	GPIO Pad Data In Register (SIUL2_GPDI454)	8	R	00h	16.2.15/ 756
16C7	GPIO Pad Data In Register (SIUL2_GPDI455)	8	R	00h	16.2.15/ 756
16C8	GPIO Pad Data In Register (SIUL2_GPDI456)	8	R	00h	16.2.15/ 756
16C9	GPIO Pad Data In Register (SIUL2_GPDI457)	8	R	00h	16.2.15/ 756
16CA	GPIO Pad Data In Register (SIUL2_GPDI458)	8	R	00h	16.2.15/ 756
16CB	GPIO Pad Data In Register (SIUL2_GPDI459)	8	R	00h	16.2.15/ 756
16CC	GPIO Pad Data In Register (SIUL2_GPDI460)	8	R	00h	16.2.15/ 756
16CD	GPIO Pad Data In Register (SIUL2_GPDI461)	8	R	00h	16.2.15/ 756
16CE	GPIO Pad Data In Register (SIUL2_GPDI462)	8	R	00h	16.2.15/ 756
16CF	GPIO Pad Data In Register (SIUL2_GPDI463)	8	R	00h	16.2.15/ 756
16D0	GPIO Pad Data In Register (SIUL2_GPDI464)	8	R	00h	16.2.15/ 756
16D1	GPIO Pad Data In Register (SIUL2_GPDI465)	8	R	00h	16.2.15/ 756
16D2	GPIO Pad Data In Register (SIUL2_GPDI466)	8	R	00h	16.2.15/ 756
16D3	GPIO Pad Data In Register (SIUL2_GPDI467)	8	R	00h	16.2.15/ 756
16D4	GPIO Pad Data In Register (SIUL2_GPDI468)	8	R	00h	16.2.15/ 756
16D5	GPIO Pad Data In Register (SIUL2_GPDI469)	8	R	00h	16.2.15/ 756
16D6	GPIO Pad Data In Register (SIUL2_GPDI470)	8	R	00h	16.2.15/ 756
16D7	GPIO Pad Data In Register (SIUL2_GPDI471)	8	R	00h	16.2.15/ 756
16D8	GPIO Pad Data In Register (SIUL2_GPDI472)	8	R	00h	16.2.15/ 756
16D9	GPIO Pad Data In Register (SIUL2_GPDI473)	8	R	00h	16.2.15/ 756
16DA	GPIO Pad Data In Register (SIUL2_GPDI474)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16DB	GPIO Pad Data In Register (SIUL2_GPDI475)	8	R	00h	16.2.15/ 756
16DC	GPIO Pad Data In Register (SIUL2_GPDI476)	8	R	00h	16.2.15/ 756
16DD	GPIO Pad Data In Register (SIUL2_GPDI477)	8	R	00h	16.2.15/ 756
16DE	GPIO Pad Data In Register (SIUL2_GPDI478)	8	R	00h	16.2.15/ 756
16DF	GPIO Pad Data In Register (SIUL2_GPDI479)	8	R	00h	16.2.15/ 756
16E0	GPIO Pad Data In Register (SIUL2_GPDI480)	8	R	00h	16.2.15/ 756
16E1	GPIO Pad Data In Register (SIUL2_GPDI481)	8	R	00h	16.2.15/ 756
16E2	GPIO Pad Data In Register (SIUL2_GPDI482)	8	R	00h	16.2.15/ 756
16E3	GPIO Pad Data In Register (SIUL2_GPDI483)	8	R	00h	16.2.15/ 756
16E4	GPIO Pad Data In Register (SIUL2_GPDI484)	8	R	00h	16.2.15/ 756
16E5	GPIO Pad Data In Register (SIUL2_GPDI485)	8	R	00h	16.2.15/ 756
16E6	GPIO Pad Data In Register (SIUL2_GPDI486)	8	R	00h	16.2.15/ 756
16E7	GPIO Pad Data In Register (SIUL2_GPDI487)	8	R	00h	16.2.15/ 756
16E8	GPIO Pad Data In Register (SIUL2_GPDI488)	8	R	00h	16.2.15/ 756
16E9	GPIO Pad Data In Register (SIUL2_GPDI489)	8	R	00h	16.2.15/ 756
16EA	GPIO Pad Data In Register (SIUL2_GPDI490)	8	R	00h	16.2.15/ 756
16EB	GPIO Pad Data In Register (SIUL2_GPDI491)	8	R	00h	16.2.15/ 756
16EC	GPIO Pad Data In Register (SIUL2_GPDI492)	8	R	00h	16.2.15/ 756
16ED	GPIO Pad Data In Register (SIUL2_GPDI493)	8	R	00h	16.2.15/ 756
16EE	GPIO Pad Data In Register (SIUL2_GPDI494)	8	R	00h	16.2.15/ 756
16EF	GPIO Pad Data In Register (SIUL2_GPDI495)	8	R	00h	16.2.15/ 756
16F0	GPIO Pad Data In Register (SIUL2_GPDI496)	8	R	00h	16.2.15/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16F1	GPIO Pad Data In Register (SIUL2_GPDI497)	8	R	00h	16.2.15/ 756
16F2	GPIO Pad Data In Register (SIUL2_GPDI498)	8	R	00h	16.2.15/ 756
16F3	GPIO Pad Data In Register (SIUL2_GPDI499)	8	R	00h	16.2.15/ 756
16F4	GPIO Pad Data In Register (SIUL2_GPDI500)	8	R	00h	16.2.15/ 756
16F5	GPIO Pad Data In Register (SIUL2_GPDI501)	8	R	00h	16.2.15/ 756
16F6	GPIO Pad Data In Register (SIUL2_GPDI502)	8	R	00h	16.2.15/ 756
16F7	GPIO Pad Data In Register (SIUL2_GPDI503)	8	R	00h	16.2.15/ 756
16F8	GPIO Pad Data In Register (SIUL2_GPDI504)	8	R	00h	16.2.15/ 756
16F9	GPIO Pad Data In Register (SIUL2_GPDI505)	8	R	00h	16.2.15/ 756
16FA	GPIO Pad Data In Register (SIUL2_GPDI506)	8	R	00h	16.2.15/ 756
16FB	GPIO Pad Data In Register (SIUL2_GPDI507)	8	R	00h	16.2.15/ 756
16FC	GPIO Pad Data In Register (SIUL2_GPDI508)	8	R	00h	16.2.15/ 756
16FD	GPIO Pad Data In Register (SIUL2_GPDI509)	8	R	00h	16.2.15/ 756
16FE	GPIO Pad Data In Register (SIUL2_GPDI510)	8	R	00h	16.2.15/ 756
16FF	GPIO Pad Data In Register (SIUL2_GPDI511)	8	R	00h	16.2.15/ 756
1700	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)	16	R/W	0000h	16.2.16/ 756
1702	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO1)	16	R/W	0000h	16.2.16/ 756
1704	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO2)	16	R/W	0000h	16.2.16/ 756
1706	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO3)	16	R/W	0000h	16.2.16/ 756
1708	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO4)	16	R/W	0000h	16.2.16/ 756
170A	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO5)	16	R/W	0000h	16.2.16/ 756
170C	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO6)	16	R/W	0000h	16.2.16/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
170E	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO7)	16	R/W	0000h	16.2.16/ 756
1710	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO8)	16	R/W	0000h	16.2.16/ 756
1712	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO9)	16	R/W	0000h	16.2.16/ 756
1714	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO10)	16	R/W	0000h	16.2.16/ 756
1716	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO11)	16	R/W	0000h	16.2.16/ 756
1718	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO12)	16	R/W	0000h	16.2.16/ 756
171A	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO13)	16	R/W	0000h	16.2.16/ 756
171C	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO14)	16	R/W	0000h	16.2.16/ 756
171E	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO15)	16	R/W	0000h	16.2.16/ 756
1720	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO16)	16	R/W	0000h	16.2.16/ 756
1722	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO17)	16	R/W	0000h	16.2.16/ 756
1724	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO18)	16	R/W	0000h	16.2.16/ 756
1726	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO19)	16	R/W	0000h	16.2.16/ 756
1728	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO20)	16	R/W	0000h	16.2.16/ 756
172A	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO21)	16	R/W	0000h	16.2.16/ 756
172C	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO22)	16	R/W	0000h	16.2.16/ 756
172E	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO23)	16	R/W	0000h	16.2.16/ 756
1730	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO24)	16	R/W	0000h	16.2.16/ 756
1732	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO25)	16	R/W	0000h	16.2.16/ 756
1734	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO26)	16	R/W	0000h	16.2.16/ 756
1736	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO27)	16	R/W	0000h	16.2.16/ 756
1738	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO28)	16	R/W	0000h	16.2.16/ 756

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
173A	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO29)	16	R/W	0000h	16.2.16/ 756
173C	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO30)	16	R/W	0000h	16.2.16/ 756
173E	Parallel GPIO Pad Data Out Register (SIUL2_PGPDO31)	16	R/W	0000h	16.2.16/ 756
1740	Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)	16	R	0000h	16.2.17/ 757
1742	Parallel GPIO Pad Data In Register (SIUL2_PGPDI1)	16	R	0000h	16.2.17/ 757
1744	Parallel GPIO Pad Data In Register (SIUL2_PGPDI2)	16	R	0000h	16.2.17/ 757
1746	Parallel GPIO Pad Data In Register (SIUL2_PGPDI3)	16	R	0000h	16.2.17/ 757
1748	Parallel GPIO Pad Data In Register (SIUL2_PGPDI4)	16	R	0000h	16.2.17/ 757
174A	Parallel GPIO Pad Data In Register (SIUL2_PGPDI5)	16	R	0000h	16.2.17/ 757
174C	Parallel GPIO Pad Data In Register (SIUL2_PGPDI6)	16	R	0000h	16.2.17/ 757
174E	Parallel GPIO Pad Data In Register (SIUL2_PGPDI7)	16	R	0000h	16.2.17/ 757
1750	Parallel GPIO Pad Data In Register (SIUL2_PGPDI8)	16	R	0000h	16.2.17/ 757
1752	Parallel GPIO Pad Data In Register (SIUL2_PGPDI9)	16	R	0000h	16.2.17/ 757
1754	Parallel GPIO Pad Data In Register (SIUL2_PGPDI10)	16	R	0000h	16.2.17/ 757
1756	Parallel GPIO Pad Data In Register (SIUL2_PGPDI11)	16	R	0000h	16.2.17/ 757
1758	Parallel GPIO Pad Data In Register (SIUL2_PGPDI12)	16	R	0000h	16.2.17/ 757
175A	Parallel GPIO Pad Data In Register (SIUL2_PGPDI13)	16	R	0000h	16.2.17/ 757
175C	Parallel GPIO Pad Data In Register (SIUL2_PGPDI14)	16	R	0000h	16.2.17/ 757
175E	Parallel GPIO Pad Data In Register (SIUL2_PGPDI15)	16	R	0000h	16.2.17/ 757
1760	Parallel GPIO Pad Data In Register (SIUL2_PGPDI16)	16	R	0000h	16.2.17/ 757
1762	Parallel GPIO Pad Data In Register (SIUL2_PGPDI17)	16	R	0000h	16.2.17/ 757
1764	Parallel GPIO Pad Data In Register (SIUL2_PGPDI18)	16	R	0000h	16.2.17/ 757

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1766	Parallel GPIO Pad Data In Register (SIUL2_PGPDI19)	16	R	0000h	16.2.17/ 757
1768	Parallel GPIO Pad Data In Register (SIUL2_PGPDI20)	16	R	0000h	16.2.17/ 757
176A	Parallel GPIO Pad Data In Register (SIUL2_PGPDI21)	16	R	0000h	16.2.17/ 757
176C	Parallel GPIO Pad Data In Register (SIUL2_PGPDI22)	16	R	0000h	16.2.17/ 757
176E	Parallel GPIO Pad Data In Register (SIUL2_PGPDI23)	16	R	0000h	16.2.17/ 757
1770	Parallel GPIO Pad Data In Register (SIUL2_PGPDI24)	16	R	0000h	16.2.17/ 757
1772	Parallel GPIO Pad Data In Register (SIUL2_PGPDI25)	16	R	0000h	16.2.17/ 757
1774	Parallel GPIO Pad Data In Register (SIUL2_PGPDI26)	16	R	0000h	16.2.17/ 757
1776	Parallel GPIO Pad Data In Register (SIUL2_PGPDI27)	16	R	0000h	16.2.17/ 757
1778	Parallel GPIO Pad Data In Register (SIUL2_PGPDI28)	16	R	0000h	16.2.17/ 757
177A	Parallel GPIO Pad Data In Register (SIUL2_PGPDI29)	16	R	0000h	16.2.17/ 757
177C	Parallel GPIO Pad Data In Register (SIUL2_PGPDI30)	16	R	0000h	16.2.17/ 757
177E	Parallel GPIO Pad Data In Register (SIUL2_PGPDI31)	16	R	0000h	16.2.17/ 757
1780	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)	32	W	0000_0000h	16.2.18/ 758
1784	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO1)	32	W	0000_0000h	16.2.18/ 758
1788	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO2)	32	W	0000_0000h	16.2.18/ 758
178C	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO3)	32	W	0000_0000h	16.2.18/ 758
1790	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO4)	32	W	0000_0000h	16.2.18/ 758
1794	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO5)	32	W	0000_0000h	16.2.18/ 758
1798	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO6)	32	W	0000_0000h	16.2.18/ 758
179C	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO7)	32	W	0000_0000h	16.2.18/ 758
17A0	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO8)	32	W	0000_0000h	16.2.18/ 758

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17A4	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO9)	32	W	0000_0000h	16.2.18/ 758
17A8	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO10)	32	W	0000_0000h	16.2.18/ 758
17AC	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO11)	32	W	0000_0000h	16.2.18/ 758
17B0	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO12)	32	W	0000_0000h	16.2.18/ 758
17B4	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO13)	32	W	0000_0000h	16.2.18/ 758
17B8	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO14)	32	W	0000_0000h	16.2.18/ 758
17BC	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO15)	32	W	0000_0000h	16.2.18/ 758
17C0	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO16)	32	W	0000_0000h	16.2.18/ 758
17C4	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO17)	32	W	0000_0000h	16.2.18/ 758
17C8	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO18)	32	W	0000_0000h	16.2.18/ 758
17CC	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO19)	32	W	0000_0000h	16.2.18/ 758
17D0	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO20)	32	W	0000_0000h	16.2.18/ 758
17D4	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO21)	32	W	0000_0000h	16.2.18/ 758
17D8	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO22)	32	W	0000_0000h	16.2.18/ 758
17DC	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO23)	32	W	0000_0000h	16.2.18/ 758
17E0	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO24)	32	W	0000_0000h	16.2.18/ 758
17E4	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO25)	32	W	0000_0000h	16.2.18/ 758
17E8	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO26)	32	W	0000_0000h	16.2.18/ 758
17EC	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO27)	32	W	0000_0000h	16.2.18/ 758
17F0	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO28)	32	W	0000_0000h	16.2.18/ 758
17F4	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO29)	32	W	0000_0000h	16.2.18/ 758
17F8	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO30)	32	W	0000_0000h	16.2.18/ 758

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17FC	Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO31)	32	W	0000_0000h	16.2.18/ 758

16.2.1 MCU ID Register #1 (SIUL2_MIDR1)

This register contains identification information about the device.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ED	PKG					0		MAJOR_MASK				MINOR_MASK			
W																
Reset	0	0*	0*	0*	0*	0*	0	0	0*	0*	0*	1*	0*	0*	0*	0*

* Notes:
- MINOR_MASK field: Value is set at factory and will vary each revision of device.
- MAJOR_MASK field: Value is set at factory and will vary each revision of device.
- PKG field: Only possible to change value by modifying the side band signals (in the Test Flash). Software can not modify. Default is 0b1000. Values are set at factory and cannot be modified.

SIUL2_MIDR1 field descriptions

Field	Description
0–15 PARTNUM	MCU Part Number-These digits identify the part number of the chip. Default is 5777.
16 ED	Read back as '1' if an ED device. Read back as '0' in PD.
17–21 PKG	Package Settings-Can be read by software to determine the package type that is used for the particular device: 0b11000: 416-pin BGA 0b11101: 512-pin BGA All other bit combinations are reserved for future use.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MAJOR_MASK	Major Mask Revision.

Table continues on the next page...

SIUL2_MIDR1 field descriptions (continued)

Field	Description
28–31 MINOR_MASK	Minor Mask Revision.

16.2.2 MCU ID Register #2 (SIUL2_MIDR2)

This register contains identification information about the device.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1				FLASH_SIZE_2			0							
W																
Reset	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FAMILYNUM								0							
W																
Reset	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0

SIUL2_MIDR2 field descriptions

Field	Description
0 SF	Manufacturer: 0 Freescale 1 STMicroelectronics
1–4 FLASH_SIZE_1	Coarse granularity for Flash memory size. Needs to be combined with FLASH_SIZE_2 to calculate the actual memory size: 0b0000: 16K 0b0001: 32K 0b0010: 64K 0b0011: 128K 0b0100: 256K 0b0101: 512K 0b0110:1024K 0b0111: 2048K 0b1000: 4096K 0b1001: 8192K 0b1010: 16384K ...

Table continues on the next page...

SIUL2_MIDR2 field descriptions (continued)

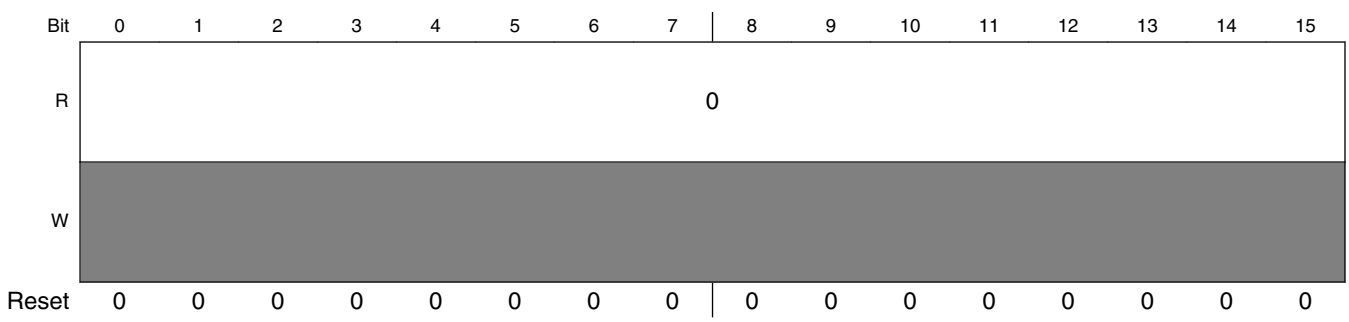
Field	Description
	0b1110: 256M 0b1111: 512M
5–8 FLASH_SIZE_2	Fine granularity for Flash memory size. Needs to be combined with FLASH_SIZE_1 to calculate the actual memory size: 0b0000: 0 x (FLASH_SIZE_1 / 8) 0b0001: 1 x (FLASH_SIZE_1 / 8) 0b0010: 2 x (FLASH_SIZE_1 / 8) ... 0b1110: 14 x (FLASH_SIZE_1 / 8) 0b1111: 15 x (FLASH_SIZE_1 / 8)
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 FAMILYNUM	ASCII character in MCU Part Number: 0x4D M All other values reserved for future use.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

16.2.3 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set irrespective of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) is enabled. The EIF bit remains set until cleared by software or through the servicing of a DMA request. The EIF bits are cleared by writing a '1' to the bits. A write of '0' has no effect.

This register contains the interrupt flags.

Address: 0h base + 10h offset = 10h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				EIF11	EIF10	EIF9	EIF8	0		EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W					w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DISR0 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 EIF11	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interruptor DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
21 EIF10	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
22 EIF9	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
23 EIF8	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIF5	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
27 EIF4	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
28 EIF3	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interruptor DMA request.

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
29 EIF2	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
30 EIF1	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
31 EIF0	External Interrupt Status Flag x-This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

16.2.4 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0). This register enables the interrupt messaging to the interrupt controller.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				EIRE11	EIRE10	EIRE9	EIRE8	0		EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

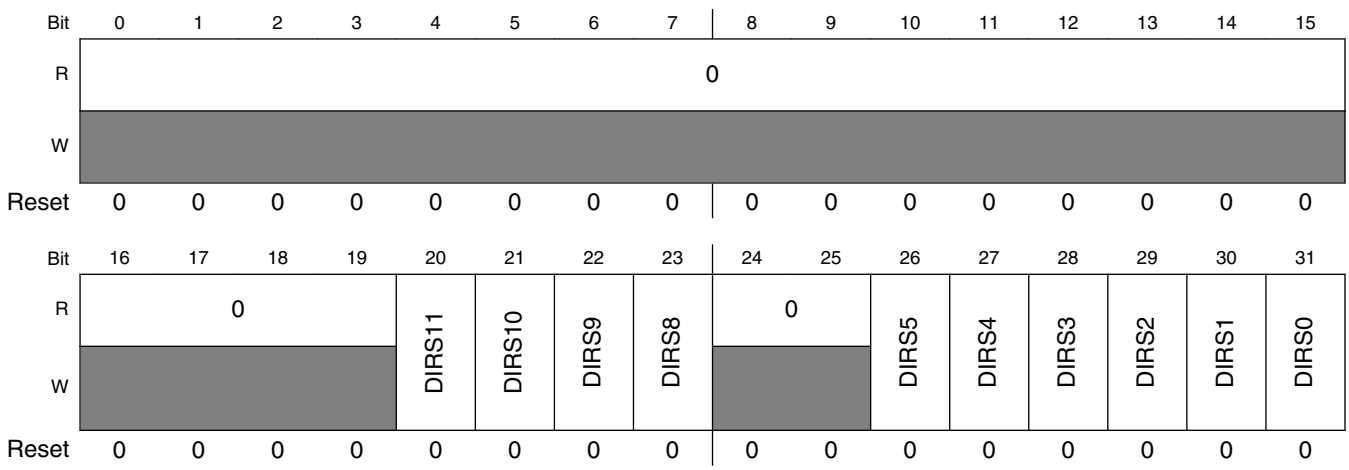
SIUL2_DIRER0 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 EIRE11	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
21 EIRE10	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
22 EIRE9	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
23 EIRE8	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIRE5	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
27 EIRE4	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
28 EIRE3	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
29 EIRE2	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
30 EIRE1	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR
31 EIRE0	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR

16.2.5 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0) and SIUL2 DMA/ Interrupt Request Enable Register0 (SIUL2_DIRER0), then the DMA/Interrupt Request Select bit determines whether DMA or an interrupt request is asserted. EIRQ are the external interrupt package pins on the device.

Address: 0h base + 20h offset = 20h



SIUL2_DIRSR0 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 DIRS11	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
21 DIRS10	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
22 DIRS9	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
23 DIRS8	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

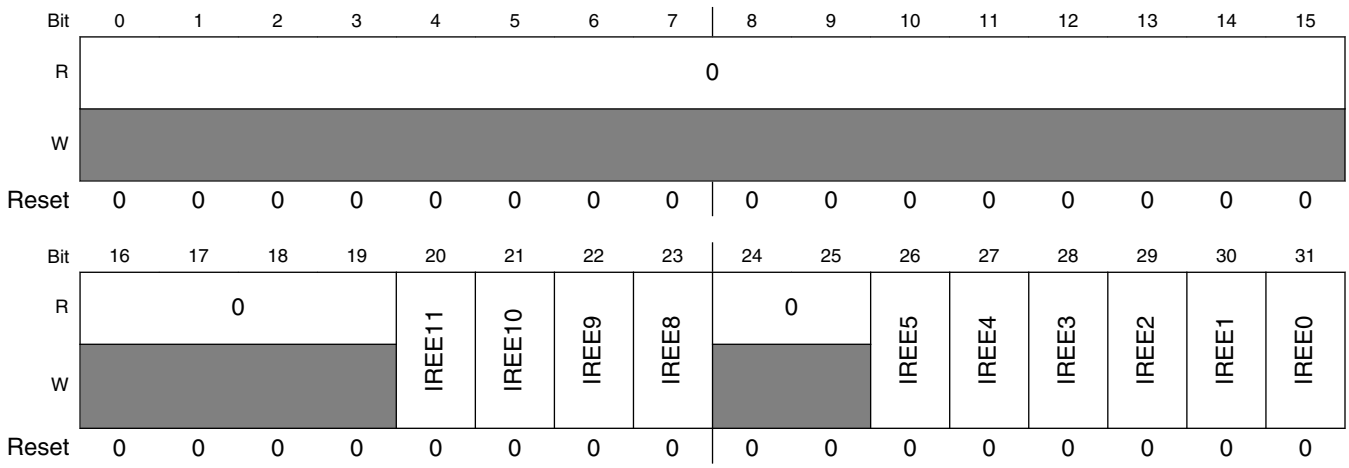
SIUL2_DIRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 DMA request is selected
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 DIRS5	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
27 DIRS4	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
28 DIRS3	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
29 DIRS2	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
30 DIRS1	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
31 DIRS0	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

16.2.6 Interrupt Rising-Edge Event Enable Register0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Address: 0h base + 28h offset = 28h



SIUL2_IREER0 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IREE11	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
21 IREE10	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 IREE9	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
23 IREE8	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 IREE5	Enable rising-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
27 IREE4	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
28 IREE3	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
29 IREE2	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 IREE1	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
31 IREE0	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

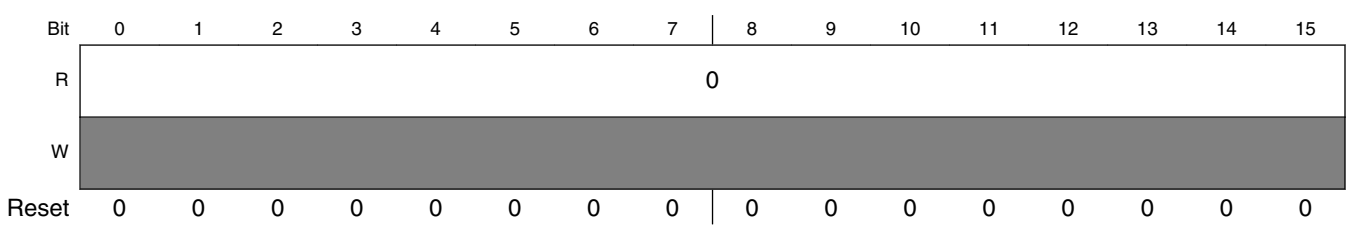
16.2.7 Interrupt Falling-Edge Event Enable Register0 (SIUL2_IFEER0)

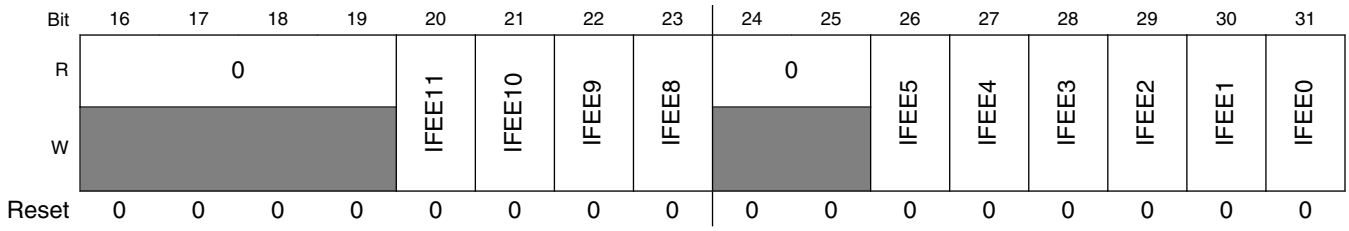
This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

NOTE

If both IREE bit and IFEE bit are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Address: 0h base + 30h offset = 30h





SIUL2_IFEER0 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IFEE11	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
21 IFEE10	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
22 IFEE9	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 IFEE8	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 IFEE5	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
27 IFEE4	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
28 IFEE3	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
29 IFEE2	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
30 IFEE1	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Table continues on the next page...

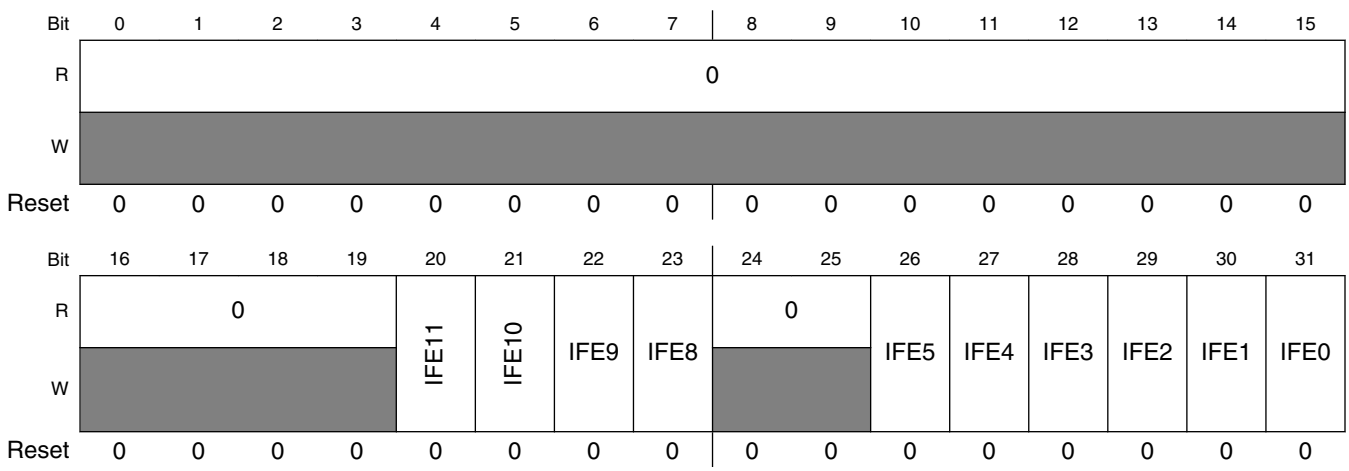
SIUL2_IFEER0 field descriptions (continued)

Field	Description
31 IFEE0	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

16.2.8 Interrupt Filter Enable Register0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

Address: 0h base + 38h offset = 38h



SIUL2_IFER0 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IFE11	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
21 IFE10	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
22 IFE9	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

Table continues on the next page...

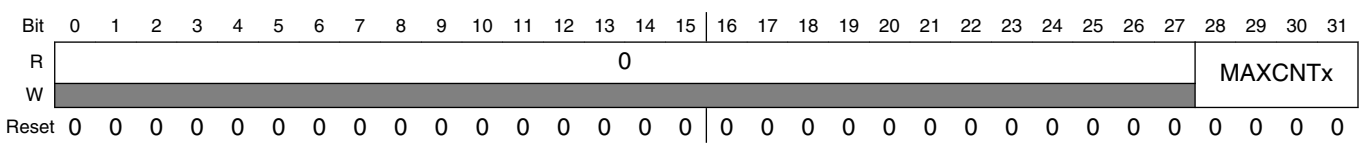
SIUL2_IFER0 field descriptions (continued)

Field	Description
23 IFE8	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 IFE5	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
27 IFE4	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
28 IFE3	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
29 IFE2	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
30 IFE1	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
31 IFE0	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

16.2.9 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

Address: 0h base + 40h offset + (4d × i), where i=0d to 31d



SIUL2_IFMCR*n* field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MAXCNTx	Maximum Interrupt Filter Counter setting Filter Period = TCK*MAXCNTx + n*TCK where (n = -1 to 3) MAXCNTx can be 0 to 15 (for MAXCNT<3 filter will behave as all PASS filter) TCK is the Prescaled Filter Clock Period, which is the IRC clock prescaled to the IFCP value specified in SIUL2_IFCPR T(IRC) Basic Filter Clock Period: 62.5 ns (F=16 MHz) Also note that Filter delay is 2 system clock cycles more than Filter period. In general, Filter Period and Filter delay are not integer multiple of TCK because input to Filter is asynchronous whereas output is synchronous with respect to the TCK. T(DELAY)=(1 to 5)*TCK + MAXCOUNT*TCK. The SW is to expect the worst filter delay value.

16.2.10 Interrupt Filter Clock Prescaler (SIUL2_IFCPR)

This register configures a clock prescaler that selects the clock for all digital filters. The prescaler is applied to the input clock to the SIUL2, which is the system PBRIDGE clock counter in the SIUL2. See the clocking chapter for information on clocking.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												IFCP			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

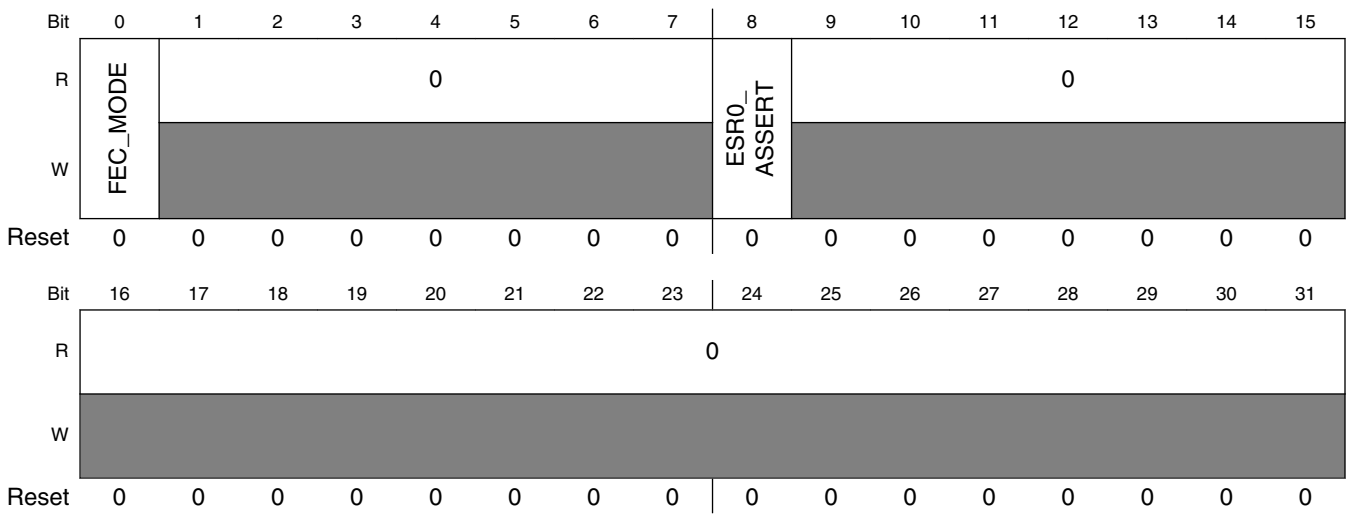
SIUL2_IFCPR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

16.2.11 SoC Configuration Register0 (SIUL2_SCR0)

The FEC_MODE bit is used to select between the Media Independent Interface (MII) and the Reduced Media Independent Interface (RMII) mode for the Fast Ethernet controller (FEC)

Address: 0h base + 100h offset = 100h



SIUL2_SCR0 field descriptions

Field	Description
0 FEC_MODE	Fast Ethernet Controller mode setting. **NOTE:** MII/RMII pad configuration is done independently in the SIUL2 MSCRs. 0 RMII is the default state 1 MII is selected.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 ESR0_ASSERT	Active high signal enabling to force ESR0 as an strong pull-down 0 ESR0 status is controlled by RG 0 ESR0 status is forced low if ESR0 is configured as output only (associated DCF record DCF.UTEST_MISCELLANEOUS[ESR0_Gate] is low)
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

16.2.12 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_*n*)

See Multiplexed Signal Configuration Registers for an introduction to the MSCRs.

This section defines the I/O pin MSCRs. These registers select the output function on the associated I/O pin.The I/O MSCRs control the drive strength, output drive circuit, pull up/down, input buffer enable, input hysteresis, input level selection, safe-mode operation, and analog input enable of the associated I/O pin.

NOTE

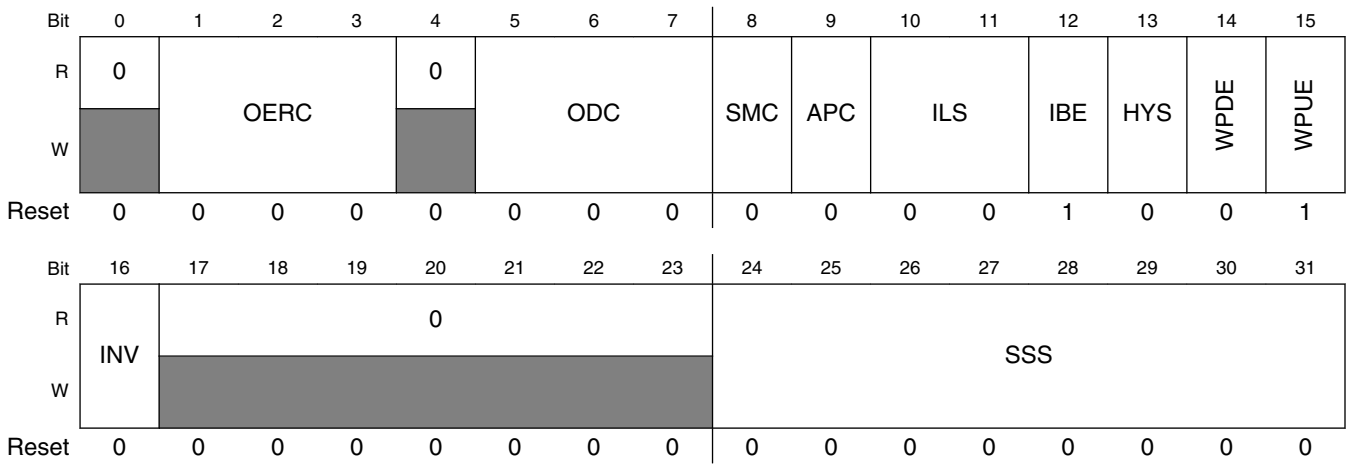
Not all pads have all bits i.e., only some pads have hysteresis. See the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

When an I/O pin is configured as a single-ended input, the input buffer must be enabled for the pin using the MSCR[IBE] bit. This enables the input to all input destinations connected to the pin, including the GPIO input. In this way, the GPIO can be read at any time, regardless of the pin function. For IP blocks that have multiple input sources, the input source is selected in the multiplexed input selection MSCRs, and the associated I/O pin MSCR for the selected input must have the MSCR[IBE] bit enabled.

All bits and fields in the I/O MSCRs are applicable for all GPIO ports on the device. This includes all ports on the device with digital output function. For ADC input ports with digital input only (no output), the OERC, ODC, SMC, INV, and SSS bits/fields do not apply. The SSS is always 8b0 for input only ports, enabling the general purpose input. The analog input path is enabled in the ADC logic when the channel is selected for conversion.

The MSCR reset state given in this section applies to all GPIO pins on the devices, which defaults to input/pull-up enabled. The exceptions to this are documented in the attached System_IO_Definition spreadsheet. Only the SMC, HYS, WPDE, and WPUE bits have exceptions to the default MSRC reset value.

Address: 0h base + 240h offset + (4d × i), where i=0d to 511d



SIUL2_MSCR_IO_*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 OERC	Output Edge Rate Control-Specifies the output impedance, drive strength, and slew rate of the associated pin. Refer to output Impedance columns in the I/O Signal Description Table for the applicable edge rates for each I/O pin on the device. The I/O Signal Description and Input Multiplexing Tables (Excel file) are attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. See the device data sheet for the electrical characteristics of the weak, medium, strong, and very strong pad types. **NOTE:** OERC[2] programming is required for forward compatibility. 000 Weak drive (800 ohm) 001 Medium drive (200 ohm) 010 Strong drive (50 ohm/10 ns) 011 Very strong drive (50 ohm/5 ns) 100 6 pF 101 12 pF 110 18 pF 111 30 pF
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 ODC	Output Drive Control-Specifies the type of output drive control for the associated pin. 000 Output buffer disabled 001 Open-drain 010 Push-pull 011 Open-source 100 Microsecond Channel LVDS 101 LFAST LVDS 110 Reserved 111 Reserved

Table continues on the next page...

SIUL2_MSCR_IO_*n* field descriptions (continued)

Field	Description
8 SMC	Safe Mode Control-Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. Resets to 0 except for PB[11] ERROR0 pin which resets to 1. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
9 APC	Analog Pad Control-Enables the pin's analog-input-path switch if the associated pin has an analog input function. 0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)
10–11 ILS	Input Level Selection-Specifies the logic family for the associated pin, which determines its logic switching levels. See the device data sheet for the electrical characteristics of the I/O pad input buffer types. 00 Automotive 01 TTL 10 LVDS 11 CMOS
12 IBE	Input Buffer Enable-Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis-Enables input hysteresis for the associated pin. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable-Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. It is OK for both WPDE and WPUE to be enabled, if they are, the pad is neither a weak pullup and a weak pulldown. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable-Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. It is OK for both WPDE and WPUE to be enabled, if they are, the pad is neither a weak pullup and a weak pulldown. 0 Disabled 1 Enabled
16 INV	Invert-The output selected by the corresponding MSCR SSS field can be inverted before it is driven on the I/O pin with the INV bit. **NOTE:** Use of the INV bit is not supported when the ODC is configured for open-drain or open-source modes, and the output data will not be as expected for an inverted open-drain or open-source connection. 0 The output selected by the SSS field is not inverted before being driven to the pin 1 The output selected by the SSS field is inverted before being driven to the pin

Table continues on the next page...

SIUL2_MSCR_IO_*n* field descriptions (continued)

Field	Description
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SSS	Source Signal Select-Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. See Chip-pin MSCR assignments and related information and Module-port MSCR assignments and SSS values.

16.2.13 Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_*n*)

See Multiplexed Signal Configuration Registers for an introduction to the MSCRs.

This section defines the multiplexed input selection MSCRs. These registers select the input source for IP blocks on the device that have inputs from multiple sources, both other IP blocks and I/O pins. It is possible to configure the pad to be in input without having to connect it externally. This is possible, either by enabling internal pull-up/pull-down or disabling the input buffer (but in this case data is seen as 0 internally).

Address: 0h base + A40h offset + (4d × i), where i=0d to 511d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0							SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_MSCR_MUX_*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 INV	Invert- The input source signal selected by the corresponding SSS field for this MSCR can be inverted before it connects to the destination input with the INV bit. 0 SSS selected input signal is connected directly to the destination input 1 SSS selected input signal is inverted before it is connected to the destination input
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SSS	Source Signal Select-Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value

Table continues on the next page...

SIUL2_MSCR_MUX_*n* field descriptions (continued)

Field	Description
	depends on the destination. See Chip-pin MSCR assignments and related information and Module-port MSCR assignments and SSS values.

16.2.14 GPIO Pad Data Out Register (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access. For mapping of ports to GPDO registers, see the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Note that the GPDO registers do not apply for ports PA[5], PA[6], and PA[7]. The PA ports are JTAG pins with no GPIO. Writes to the GPDO register corresponding to these ports have no effect.

Address: 0h base + 1300h offset + (1d × i), where i=0d to 511d

Bit	0	1	2	3	4	5	6	7
Read	0							PDO
Write								
Reset	0	0	0	0	0	0	0	0

SIUL2_GPDO*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDO	Pad Data Out-This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO padwhen the pad is configured as an output

16.2.15 GPIO Pad Data In Register (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access. For mapping of ports to GPDI registers, See the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.Note that the GPDI registers do not apply for ports PA[5], PA[6], and PA[7]. These three PA ports are JTAG pins with no GPIO. The GPDI values corresponding to these ports always read a value of zero.

Address: 0h base + 1500h offset + (1d × i), where i=0d to 511d

Bit	0	1	2	3	4	5	6	7
Read	0							PDI
Write								
Reset	0	0	0	0	0	0	0	0

SIUL2_GPDI*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDI	Pad Data In-This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

16.2.16 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write. Note that the PGPDO registers do not apply for ports PA[5], PA[6], and PA[7]. These ports are JTAG pins with no GPIO. Writes to the PGPDO register corresponding to these ports have no effect.

NOTE

The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

Address: 0h base + 1700h offset + (2d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	PPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDO*n* field descriptions

Field	Description
0–15 PPDO	Parallel Pad Data Out-Write or read the data register that stores the value to be driven on the pad in output mode. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO). The x and y bit indices define which PPDO register bit is equivalent to which PDO register bit, according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

16.2.17 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read. Note that the GPDI registers do not apply for ports PA[5], PA[6], and PA[7]. These ports are JTAG pins with no GPIO. The GPDI values corresponding to these ports always read a value of zero

Address: 0h base + 1740h offset + (2d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	PPDI															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDI*n* field descriptions

Field	Description
0–15 PPDI	Parallel Pad Data In -Reads the current pad value. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and y bit indices define which PPDI register bit is equivalent to which PDI register bit, according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

16.2.18 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x] [0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0. Note that the MPGPDO registers do not apply for ports PA[5], PA[6], and PA[7]. These are JTAG pins with no GPIO. Writes to the MPGPDO register corresponding to these ports have no effect

Address: 0h base + 1780h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W	MASK																MPPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_MPGPDO*n* field descriptions

Field	Description
0–15 MASK	Mask Field-Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0 The associated bit value in the MPPDO[x] field is ignored 1 The associated bit value in the MPPDO[x] field is written
16–31 MPPDO	Masked Parallel Pad Data Out-Write the data register that stores the value to be driven on the pad in output mode. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Output Registers (PDO). The x and bit indices define which MPPDO register bit is equivalent to which PDO register bit, according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

16.2.19 Multiplexed Signal Configuration Registers

The MSCRs control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR register for each I/O pin on the

device. Each MSCR is independent of the others, except in the case of an LVDS configuration, where two MSCRs must be configured together in this mode. When LVDS is configured, cmos pad is disabled (ibe and obe=0) and when LVDS is not configured, (either of two MSCRs not configured for lvds) then the lvds pad is disabled (lvds_ibe and lvds_obe=0).

The MSCRs are divided into two sets:

- I/O pin control registers, see I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_*n*).
- Multiplexed input selection registers Multiplexed Signal Configuration Register for Multiplexed Input Selection (SIUL2_MSCR_MUX_*n*).

The signal data flow from IP blocks on the device to and from the I/O pins, and from IP block to IP block is given in the following figure.

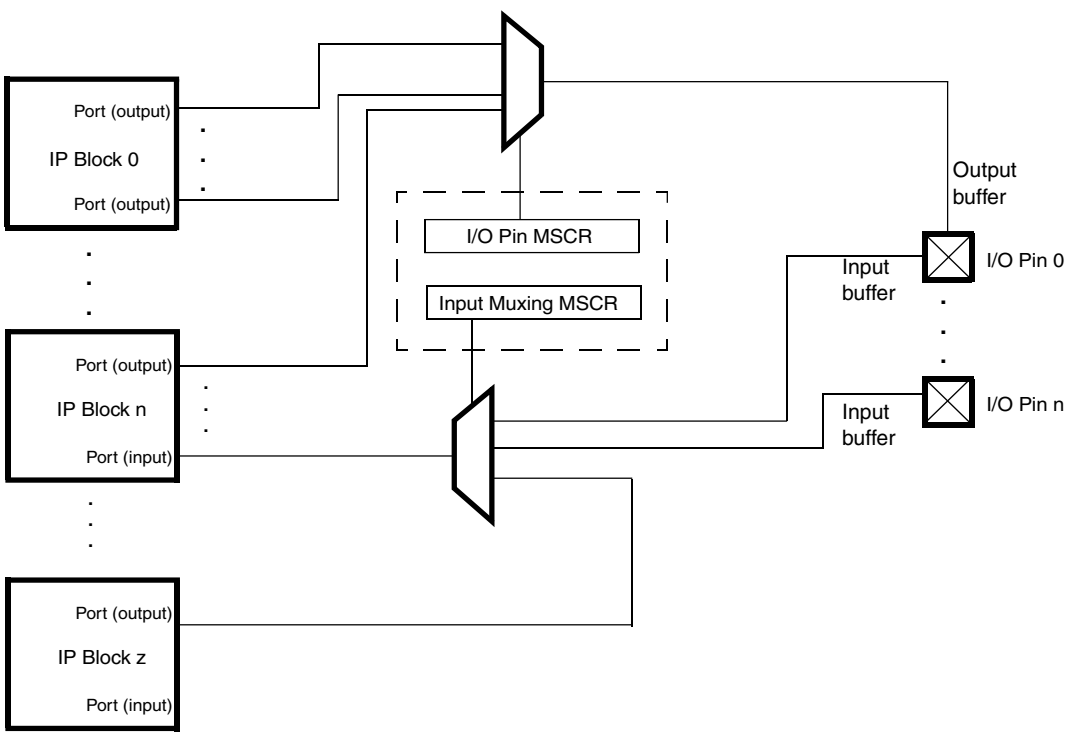


Figure 16-2. MSCR I/O Pin and IP Block Port Connectivity

16.2.20 Chip-pin MSCR assignments and related information

For device pin MSCR definitions, see the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

16.2.21 Module-port MSCR assignments and SSS values

For each internal module port that is or can be configured as an input and has more than one possible source. See the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the Input Multiplexing Table tab. The Input Multiplexing Table shows the assigned MSCR, and the SSS values and their corresponding source signals.

16.3 Functional description

16.3.1 General

This section provides a complete functional description of the SIUL2.

16.3.2 Pad control

The SIUL2 is capable of controlling the electrical characteristic of 341 pads on this device. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2. Not all of these features will be supported on all devices or on all pads as this is dependent on the specific application needs and the pads implemented on the device.

The required controls are:

- Slew rate control
  - This is needed to offer improved EMC performance.
  - Support for a fast and a slow slew rate.
- Output Impedance Control

    - 4 output impedances supported
- Drive strength control for EBI pins
    - This is needed to offer improved EMC performance.
    - Up to 4 different drive strength levels can be supported.
- Internal weak pull capability
    - This is needed to offer flexibility in the device configuration and the elimination of external hardware in some cases.
    - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Control of analog path switches
- Safe Mode behavior configuration
- Open drain/source enable
    - Needed to support different pads muxed (e.g. mux IIC, with non open drain pads)
- Pin Function Assignment
    - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

16.3.3 General purpose input or output pads (GPIO)

The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage 341 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in the following figure, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

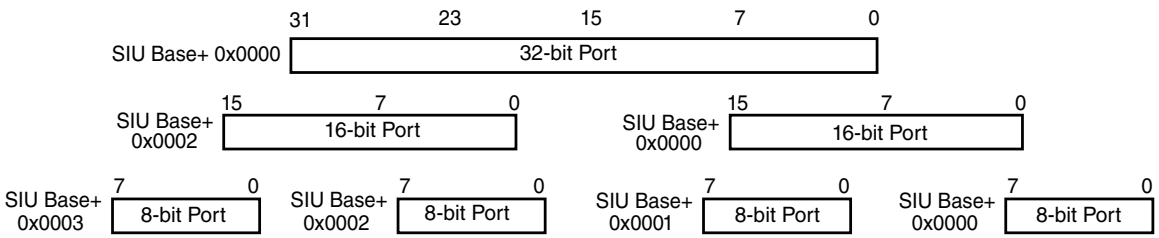


Figure 16-3. Data Port example arrangement showing configuration for different port width accesses

This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

All general purpose pads are implemented as bidirectional.

Note

In case the bi-directional operation impacts some performances (e.g. ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to "Input Only".

16.3.4 External interrupts/DMA requests (EIRQ pins)

The SIUL2 supports nine external interrupts that are allocated to EIRQ pins on the device.

The SIUL2 supports two interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support eight external interrupt sources from the device pads.

Refer to the following figure for an overview of the external interrupt implementation.

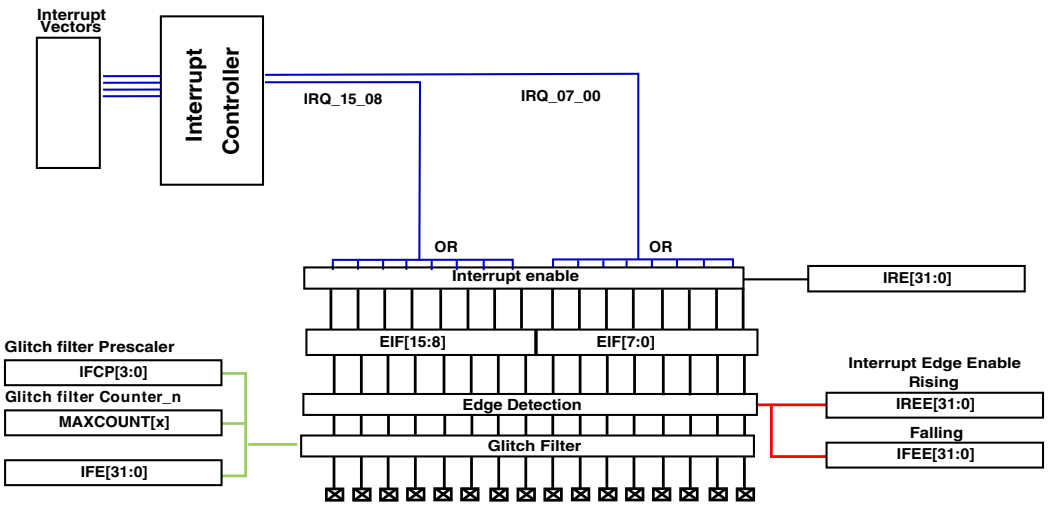


Figure 16-4. External interrupt pad diagram

All the external interrupt pads within a single group (max 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU specific number of external interrupt lines can have a digital glitch filters applied to them. The supported range is 1 to 8. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on an interrupt line.

16.3.5 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

1. Mask interrupts by clearing the EIREn bits in DIRER0.
2. Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
3. Configure the appropriate bits in the MSCR[0–511] register for the external interrupt pin(s) desired as follows:
   a. Clear the ODC bits to do disable output.
   b. Set the IBE bit to enable the pin's input buffer.
   c. If using the internal weak pullup/pulldown, configure the appropriate WPUE and WPDE bits.

Note

MSCR_SSS bits do not need to be changed for either MSCR[0:511] or MSCR[512:1023] since the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

1. Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.
2. Select the desired glitch filter setup for the pins by writing the following:
   a. Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
   b. Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR.
   c. Then enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.

Note

The glitch filter clock runs on the internal RC oscillator clock whereas the SIUL logic runs off of the PBRIDGEx clock. The IRC clock is at a fixed frequency of 16 MHz. The PBRIDGEx clock can run as high as 50 MHZ when core clock is at 300 MHz.

The glitch filter is enabled on IRC clock, so make sure at least one IRC clock occurs after enabling it before trying to detect glitchless events.

3. Write to EIFn bits in DISR0 to as desired to clear any flags (For DMA, its self clearing).
4. Enable the interrupt pins by setting the appropriate EIREn bits in DIREER0.

16.3.6 External interrupt management

Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0—Figure 5). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured.

The active IRQ edge is controlled by the users through the configuration of the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag that is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

This device supports two SIUL2 interrupt vectors aggregating a total of nine external interrupt sources as shown in the following figure.

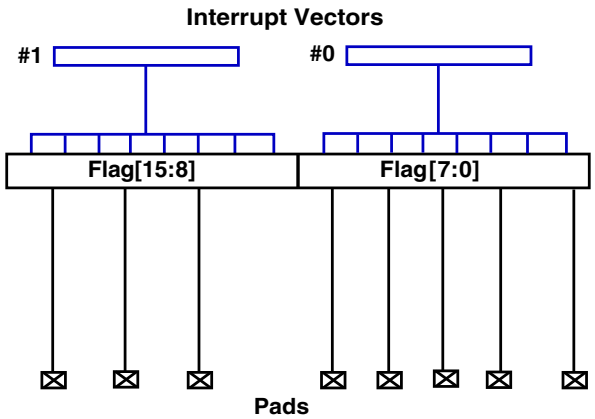


Figure 16-5. Interrupt to vector mapping at MCU level

16.3.7 External interrupt request

The EIRQ input pins on the device are sources for interrupt or DMA requests. The EIRQ pins for the device map to two interrupt requests. The mapping of interrupt requests and EIRQ pins is given in the following table.

Table 16-1. Interrupt source mapping to SIUL2 interrupt request output

Interrupt requests	EIRQ pins
0	EIRQ0 | EIRQ1 | EIRQ2 | EIRQ3 | EIRQ4 | EIRQ5
1	EIRQ8 | EIRQ9 | EIRQ10 | EIRQ11

16.3.8 Interrupt Vector

The EIRQ pins on the device map to an Interrupt Vector in the Interrupt Controller.

Note

See the Chapter 7, Device Configuration, Table 42 for Interrupt Vector table, DMA requests.

The EIRQ pins on the device map to an independent DMA request channel in the DMA controller.

Note

See the Chapter 7, Device Configuration, Table 45 for DMA request mapping table.

Chapter 17
Crossbar Switch (XBAR)

17.1 Introduction

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

17.1.1 Features

The crossbar switch includes these distinctive features:

- Symmetric crossbar bus switch implementation
- • Concurrent accesses from different masters to different slaves
  • Configurable slave arbitration attributes on a slave-by-slave basis
- 64-bit datapath width
- Support for 8-, 16-, 32-, and 64-bit single transfers
- Support for a variety of 4-, 8-, and 16-beat burst transfers including a 4-beat, 64-bit burst for cache line accesses
- Operation at one-to-one clock frequency with the bus masters
- Support for low-power park mode

17.2 Memory map and register definition

Each slave port of the crossbar switch contains configuration registers. Read and write transfers of the configuration registers require two bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write. The core, for example, takes a data storage interrupt.

NOTE

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the Chip Configuration details for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch registers are based on the physical port connections, not the logical port numbers.

XBAR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XBAR Priority Registers Slave (XBAR_PRS0)	32	R	See section	17.2.1/769
10	XBAR Control Register (XBAR_CRS0)	32	R/W	00FF_0000h	17.2.2/771
100	XBAR Priority Registers Slave (XBAR_PRS1)	32	R	See section	17.2.1/769
110	XBAR Control Register (XBAR_CRS1)	32	R/W	00FF_0000h	17.2.2/771
200	XBAR Priority Registers Slave (XBAR_PRS2)	32	R	See section	17.2.1/769
210	XBAR Control Register (XBAR_CRS2)	32	R/W	00FF_0000h	17.2.2/771
300	XBAR Priority Registers Slave (XBAR_PRS3)	32	R	See section	17.2.1/769
310	XBAR Control Register (XBAR_CRS3)	32	R/W	00FF_0000h	17.2.2/771
400	XBAR Priority Registers Slave (XBAR_PRS4)	32	R	See section	17.2.1/769
410	XBAR Control Register (XBAR_CRS4)	32	R/W	00FF_0000h	17.2.2/771
500	XBAR Priority Registers Slave (XBAR_PRS5)	32	R	See section	17.2.1/769
510	XBAR Control Register (XBAR_CRS5)	32	R/W	00FF_0000h	17.2.2/771
600	XBAR Priority Registers Slave (XBAR_PRS6)	32	R	See section	17.2.1/769
610	XBAR Control Register (XBAR_CRS6)	32	R/W	00FF_0000h	17.2.2/771
700	XBAR Priority Registers Slave (XBAR_PRS7)	32	R	See section	17.2.1/769
710	XBAR Control Register (XBAR_CRS7)	32	R/W	00FF_0000h	17.2.2/771

17.2.1 XBAR Priority Registers Slave (XBAR_PRS*n*)

NOTE

See Chip Configuration section for this device's reset values.

The priority registers (XBAR*x*_PRS*n*) set the priority of each master port on a per-slave-port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. The XBAR*x*_PRS*n* register is read-only; attempts to write to it have no effect on it and result in a bus-error response to the master initiating the write.

See the "Crossbar switch configuration" section in the chip-specific details for XBAR*x*_PRS*n*priority values.

Address: FC00_4000h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	M7			0	M6			0	M5			0	M4		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

XBAR_PRS*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 M7	Master 7 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 M6	Master 6 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port.

Table continues on the next page...

XBAR_PRS*n* field descriptions (continued)

Field	Description
	011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 M5	Master 5 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 M4	Master 4 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 M3	Master 3 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 M2	Master 2 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port.

Table continues on the next page...

XBAR_PRS*n* field descriptions (continued)

Field	Description
	010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 priority. Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

17.2.2 XBAR Control Register (XBAR_CRS*n*)

Address: FC00_4000h base + 10h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HRP	0						HPE7	HPE6	HPE5	HPE4	HPE3	HPE2	HPE1	HPE0
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						ARB		0		PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBAR_CRS*n* field descriptions

Field	Description
0 RO	Read only Forces both of the slave port's registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writable. 1 The slave port's registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HRP	Halt Request Priority Determines if a request to halt the crossbar is treated as the highest priority request, or the lowest priority request. 0 The halt request has the highest priority for arbitration on this slave port. 1 The halt request has the lowest initial priority for arbitration on this slave port.
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 HPE7	High Priority Enable Determines if Master 7 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 7 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 7 can temporarily elevate the priority level of the master's request to the slave.
9 HPE6	High Priority Enable Determines if Master 6 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 6 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 6 can temporarily elevate the priority level of the master's request to the slave.
10 HPE5	High Priority Enable Determines if Master 5 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 5 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 5 can temporarily elevate the priority level of the master's request to the slave.
11 HPE4	High Priority Enable Determines if Master 4 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 4 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 4 can temporarily elevate the priority level of the master's request to the slave.
12 HPE3	High Priority Enable

Table continues on the next page...

XBAR_CRS*n* field descriptions (continued)

Field	Description
	Determines if Master 3 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 3 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 3 can temporarily elevate the priority level of the master's request to the slave.
13 HPE2	High Priority Enable Determines if Master 2 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 2 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 2 can temporarily elevate the priority level of the master's request to the slave.
14 HPE1	High Priority Enable Determines if Master 1 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 1 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 1 can temporarily elevate the priority level of the master's request to the slave.
15 HPE0	High Priority Enable Determines if Master 0 is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master 0 are ignored and are treated the same as regular requests. 1 High Priority requests from Master 0 can temporarily elevate the priority level of the master's request to the slave.
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking control Determines the slave port's parking control. The low-power park feature results in overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field.

Table continues on the next page...

XBAR_CRS*n* field descriptions (continued)

Field	Description
	01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port. 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state. 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **NOTE:** Only select master ports that are actually present on the device. If not, undefined behavior may occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

17.3 Functional description

This section provides the functional details of the crossbar switch.

17.3.1 General operation

When a master sends an access to the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. It is possible to make single-clock (zero wait state) accesses through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Since the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply enters a wait state.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
- An outstanding request to one slave port that has a long response time and
- A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes. When a master has control of a given slave port, the crossbar returns all response information from the slave back to the requesting master.

The crossbar terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus even though a default master may be granted access to the slave port.

When a slave bus is being IDLEd by the crossbar, it can park the slave port on the master port indicated by the XBAR*x*_CRS*n*[PARK]. This is done in an attempt to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low-power park mode in attempt to save power by using XBAR*x*_CRS*n*[PCTL].

17.3.2 Register coherency

Since the content of the registers has a real-time effect on the operation of the crossbar, it is important to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave-port-related master accesses; instead, they track only with slave accesses.

17.3.3 Arbitration

The crossbar switch supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

17.3.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the XBAR*x*_PRS*n* (priority registers). If two masters request access to a slave port, the master with the higher priority in the selected priority register gains control over the slave port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port performs an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed-length burst transfer or a locked transfer. In this case, the new requesting master must wait until the end of the burst transfer or locked transfer before it is granted control of the slave port.

If the new requesting master's priority level is lower than the master that currently has control of the slave port, the new requesting master is forced to wait until the current master runs one of the following cycles:

- An IDLE cycle
- A non-IDLE cycle to a location other than the current slave port

17.3.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the physical master port number. This relative priority is compared to the ID (master port number) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far the ID of the requesting master is ahead of the ID of the last master.

After access is granted to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order 4, 5, and then 0.

Parking may continue to be used in a round-robin mode, but it does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

17.3.3.3 Priority assignment

Each master port needs to be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (XBAR*x*_PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

17.4 Initialization/application information

No initialization is required by or for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. Settings and priorities should be programmed to achieve maximum system performance.

Chapter 18
Crossbar Integrity Checker (XBIC)

18.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers. For forward signals (master to slave), it is done by verifying the integrity of the attribute information using an 8-bit Error Detection Code (EDC). The EDC detects any single- or double-bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. For feedback signals (slave to master), it is done by comparing the consistency of the signals during the AHB dataphase.There are three signals from slave to master, hready, hresp0, and hresp2. If any of the master signals is different from the slave signals during dataphase, the error will be reported in the Error Status Register.

During the address phase of the system bus pipelined protocol, the XBIC generates an 8-bit EDC checkbit value for every transfer, passing these checkbits through the crossbar to the targeted slave where it is compared against the 8-bit EDC checkbits calculated from the crossbar slave attributes. The EDC is calculated for 64 bits of attribute information: 32 bits of system bus attribute control plus 32 bits associated with the processor core's decorated storage attribute. The resulting (72,64) code is a distance 4 encoding using a minimum odd weight H matrix definition. The EDC definition has been optimized for timing considerations with the system bus attribute signals exclusively using weight 3 code values and the decoration attribute using a mix of weight 3 and 5 codes.

During the data phase of the system bus pipelined protocol, the XBIC compares the three signals of hready, hresp0, hresp2 from entering the crossbar on the slave side to leaving the crossbar on the master side. Data phase is often more than one clock cycle due to wait cycles. All three signals are compared clock by clock during the whole data cycle.

18.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
  - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
  - hready, hresp0, and hresp2 are compared from entering the crossbar on the slave side to leaving the crossbar on the master side
- Error injection for testing
  - Programmable master and slave port specifiers
  - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
  - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

18.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in Figure 18-1 and Figure 18-2.

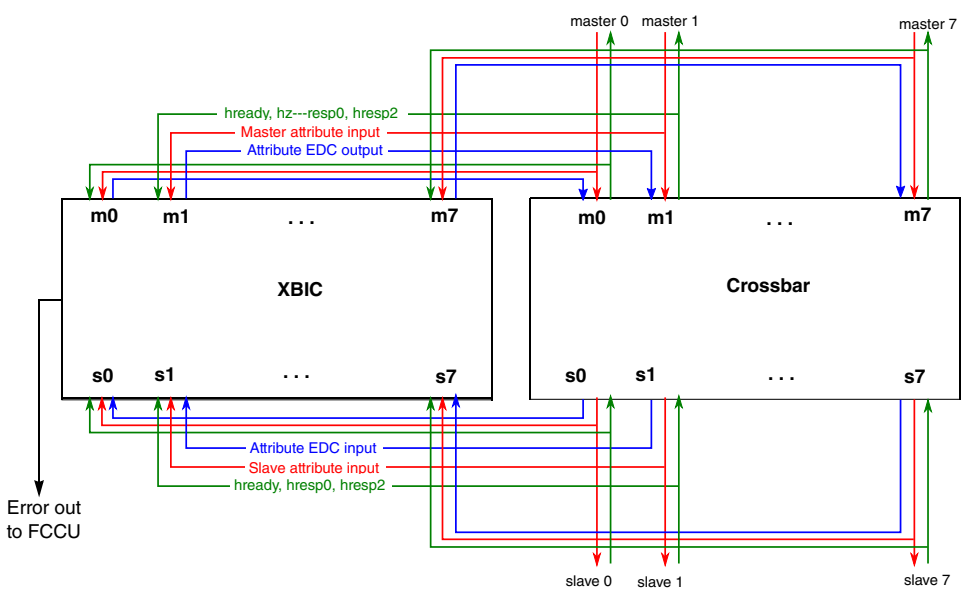


Figure 18-1. XBIC system block diagram

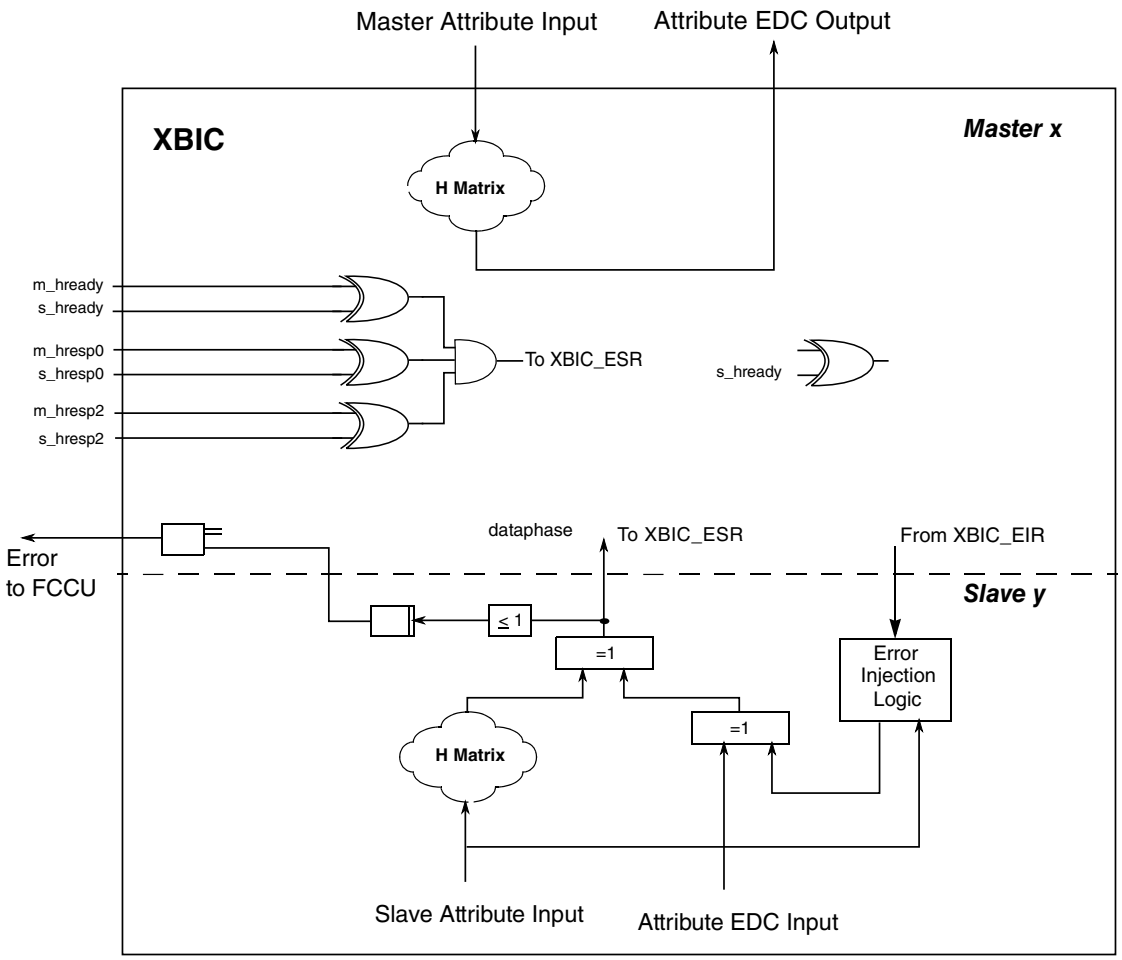


Figure 18-2. XBIC block diagram

18.4 External signal description

The XBIC has no external interface signals.

18.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

XBIC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XBIC Module Control Register (XBIC_MCR)	32	R/W	FFFF_0000h	18.5.1/782
4	XBIC Error Injection Register (XBIC_EIR)	32	R/W	0000_0000h	18.5.2/784
8	XBIC Error Status Register (XBIC_ESR)	32	R	0000_0000h	18.5.3/784
C	XBIC Error Address Register (XBIC_EAR)	32	R	0000_0000h	18.5.4/787

18.5.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 0. 1 Attribute integrity checking enabled for slave port 0.
1 SE1	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 1. 1 Attribute integrity checking enabled for slave port 1.
2 SE2	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 2. 1 Attribute integrity checking enabled for slave port 2.
3 SE3	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 3. 1 Attribute integrity checking enabled for slave port 3.
4 SE4	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 4. 1 Attribute integrity checking enabled for slave port 4.
5 SE5	Slave Port Enable for EDC error detect.

Table continues on the next page...

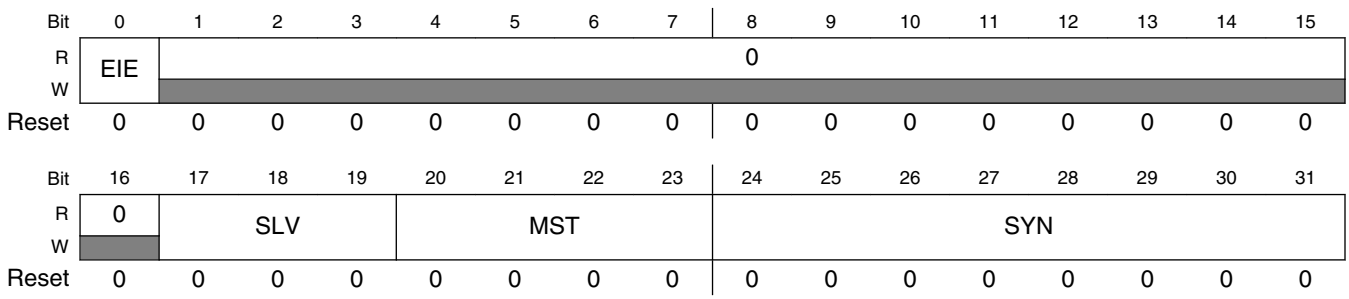
XBIC_MCR field descriptions (continued)

Field	Description
	0 Attribute integrity checking disabled for slave port 5. 1 Attribute integrity checking enabled for slave port 5.
6 SE6	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 6. 1 Attribute integrity checking enabled for slave port 6.
7 SE7	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 7. 1 Attribute integrity checking enabled for slave port 7.
8 ME0	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 0. 1 Attribute integrity checking enabled for master port 0.
9 ME1	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 1. 1 Attribute integrity checking enabled for master port 1.
10 ME2	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 2. 1 Attribute integrity checking enabled for master port 2.
11 ME3	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 3. 1 Attribute integrity checking enabled for master port 3.
12 ME4	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 4. 1 Attribute integrity checking enabled for master port 4.
13 ME5	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 5. 1 Attribute integrity checking enabled for master port 5.
14 ME6	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 6. 1 Attribute integrity checking enabled for master port 6.
15 ME7	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 7. 1 Attribute integrity checking enabled for master port 7.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.5.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Address: 0h base + 4h offset = 4h



XBIC_EIR field descriptions

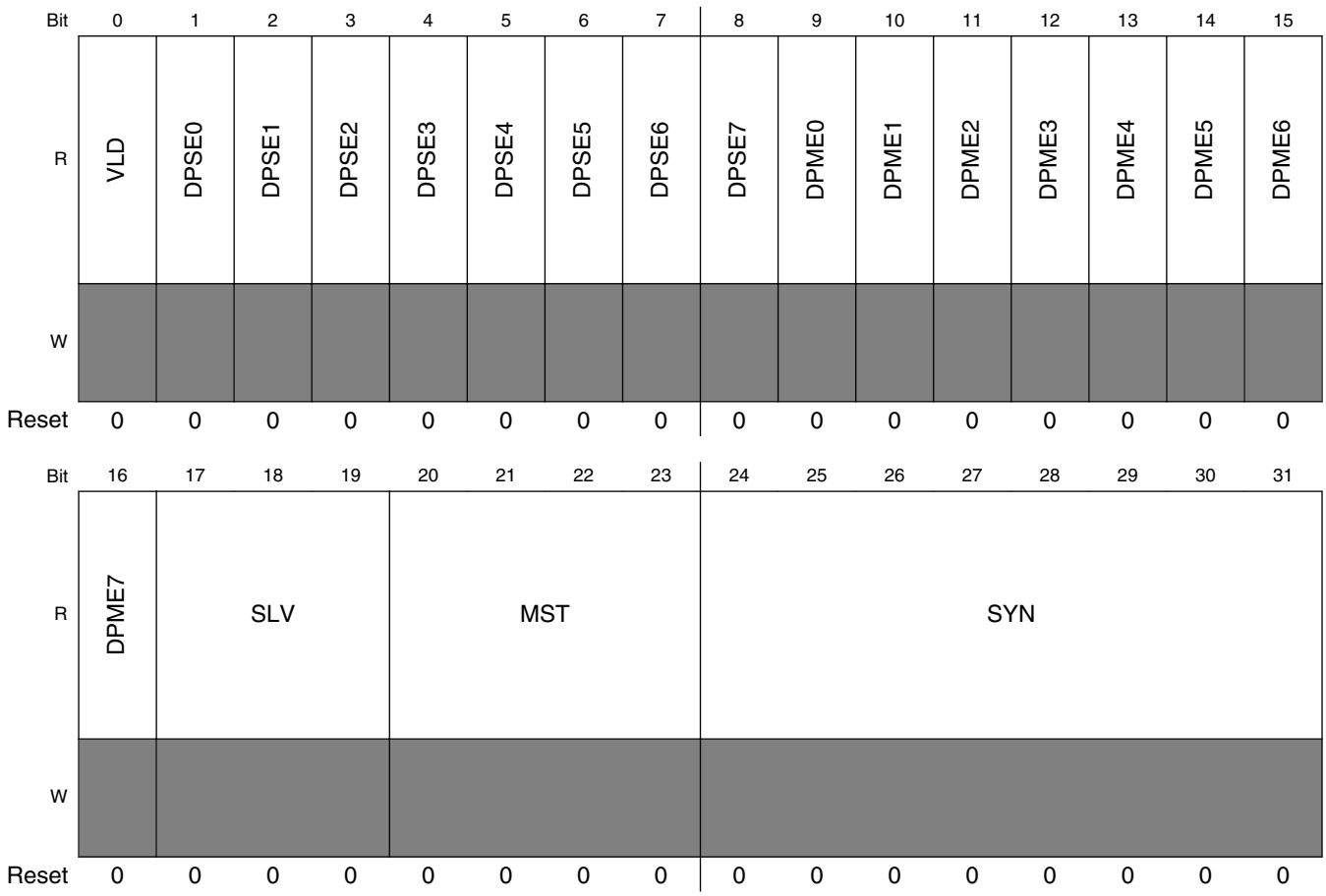
Field	Description
0 EIE	Error Injection Enable. 0 Error injection disabled 1 Error injection enabled
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 SLV	Target Slave Port. Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20–23 MST	Target Master ID. Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24–31 SYN	Syndrome. This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

18.5.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the last transfer. If an attribute integrity check error was detected the slave port, the master port, and the syndrome are captured in the SLV, MST, and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are catured in the DPSE0-7 and DPME0-7 fields.

This register is cleared only on reset.

Address: 0h base + 8h offset = 8h



XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid. 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error. 0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error. 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error. 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error. 0 No error on slave port 3 1 Data phase error on slave port 3

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
5 DPSE4	Data phase slave port error. 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error. 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error. 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error. 0 No error on slave port 7 1 Data phase error on slave port 7
9 DPME0	Data phase master port error. 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error. 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error. 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error. 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error. 0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error. 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error. 0 No error on master port 6 1 Data phase error on master port 6
16 DPME7	Data phase master port error. 0 No error on master port 7 1 Data phase error on master port 7

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
17–19 SLV	Slave Port. The slave port targeted by the last transfer with an error.
20–23 MST	Master ID. The master ID value of the last transfer with an error.
24–31 SYN	Syndrome. The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see Table 18-1 .

18.5.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the last transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_EAR field descriptions

Field	Description
0–31 ADDR	The address of the last transfer with an error.

18.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the attribute information for crossbar transfers using an 8-bit Error Detection Code (EDC). The EDC detects any single or double bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a per-slave-port basis. The EDC parity is generated by the XBIC for each master port during the system bus address phase pipeline stage, routed through the crossbar via a sideband signal, and checked against the slave attribute EDC generated for each enabled slave port during the first system bus data phase cycle. Detection of an error does *not* generate a bus error. The XBIC attribute integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

The XBIC can be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified *without* compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared.

Forwarding errors (master to slave) are detected during the AHB address phase. When a forwarding error is detected, due either to a hardware fault or error injection, information related to the error is captured and reported in the Error Status Register (XBIC_ESR) and Error Address Register (XBIC_EAR). The XBIC_ESR and XBIC_EAR registers contain information about the *last* transfer with a detected error and are only cleared on reset. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in Table 18-1. Any other syndrome indicates a multi-bit error.

XBIC also detects backward signal (slave to master) errors during AHB data phase. For the three slave driven signals (hready, hresp0, and hresp2), XBIC performs the comparison between the input of the slave port and output of the master port during the AHB data phase. When a mismatch occurs, the error will be captured. The current slave port will be registered in XBIC_ESR bit field DPSE0-7, and the current master port will be registered in XBIC_ESR bit field DSME0-7.

Table 18-1. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f

Table continues on the next page...

Table 18-1. Hexadecimal attribute single-bit error syndromes (continued)

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

The H matrix for the attribute EDC is shown in Table 18-2.

Table 18-2. XBIC H Matrix Definition

Checkbits [7:0]		Attribute Bit																															
		hwrite	htrans[0]	hsize[2]	hsize[1]	hsize[0]	hprot[5]	hprot[4]	hprot[3]	hprot[2]	hprot[1]	hprot[0]	hburst[2]	hburst[1]	hburst[0]	hmastlock	hunalign	hbstrb[7]	hbstrb[6]	hbstrb[5]	hbstrb[4]	hbstrb[3]	hbstrb[2]	hbstrb[1]	hbstrb[0]	hmaster[3]	hmaster[2]	hmaster[1]	hmaster[0]	hslave[2]	hslave[1]	hslave[0]	hdecorated
7												*	*								*				*	*	*	*	*	*	*	*	*
6														*				*		*		*	*	*				*	*	*	*	*	*
5													*	*	*	*	*	*	*		*				*	*	*						*
4						*	*	*	*	*	*	*						*	*	*							*					*	
3			*	*	*				*	*	*				*	*	*				*			*							*		
2		*		*	*		*	*			*			*			*						*	*	*					*			
1		*	*		*	*		*		*						*			*	*		*				*			*				
0		*	*	*		*	*		*			*	*		*							*	*					*					

Checkbits [7:0]		Attribute Bit																															
		hdecor[31]	hdecor[30]	hdecor[29]	hdecor[28]	hdecor[27]	hdecor[26]	hdecor[25]	hdecor[24]	hdecor[23]	hdecor[22]	hdecor[21]	hdecor[20]	hdecor[19]	hdecor[18]	hdecor[17]	hdecor[16]	hdecor[15]	hdecor[14]	hdecor[13]	hdecor[12]	hdecor[11]	hdecor[10]	hdecor[9]	hdecor[8]	hdecor[7]	hdecor[6]	hdecor[5]	hdecor[4]	hdecor[3]	hdecor[2]	hdecor[1]	hdecor[0]
7				*	*	*	*	*	*	*		*	*	*								*	*	*				*					
6			*	*							*				*		*		*	*	*	*	*	*		*		*				*	*
5		*	*													*		*			*	*	*	*	*		*	*	*	*	*		*
4												*	*	*		*			*	*				*	*	*	*	*	*	*	*		
3			*					*	*	*	*			*			*							*	*	*			*	*	*	*	*
2		*		*		*	*			*			*		*	*				*			*				*			*	*	*	*
1				*	*		*		*			*			*		*	*				*	*				*		*		*	*	*
0		*		*	*	*		*			*							*	*		*	*					*	*	*	*		*	

Chapter 19
Peripheral Bridge (PBRIDGE)

19.1 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access a majority of peripherals on the device.

The peripheral bridge occupies a 64 MB portion of the address space. (Not all peripheral slots may be used. See the Device Configuration chapter or the Memory Map chapter for details on slot assignment.) The bridge includes separate clock enable inputs for each of the slots, to accommodate slower peripherals.

19.1.1 Fetures

Key features of the peripheral bridge are:

- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Each independently configurable peripheral includes a clock enable, which allows peripherals to operate at any speed less than the system clock rate
- Programming model that provides memory protection functionality

19.1.2 General operation

The slave devices connected to the peripheral bridge are modules that contain readable/writable control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the peripherals. The peripheral bridge captures read data from the peripheral interface and drives it to the crossbar switch.

NOTE

FlexRay access to peripherals via the peripheral bridge is not supported.

Each peripheral is typically allocated one 16-KB block (or slot) of the memory map.

Usually, the peripheral bridge uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (dynamic bus sizing) is performed. However, there are the following exceptions:

- Aligned 64-bit reads are permitted to 32-bit slots.
- Aligned 64-bit writes may be targeted to 32-bit slots.
- Aligned or misaligned 64-bit instruction fetches may also be performed to 32-bit slots.

All other 64-bit accesses result in an error response.

19.2 Memory map/register definition

PBRIDGE memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Privilege Register A (PBRIDGE_MPRA)	32	R/W	7777_7777h	19.2.1/794
4	Master Privilege Register B (PBRIDGE_MPRB)	32	R/W	7777_7777h	19.2.2/795
100	Peripheral Access Control Register (PBRIDGE_PACRA)	32	R/W	4444_4444h	19.2.3/796
104	Peripheral Access Control Register (PBRIDGE_PACRB)	32	R/W	4444_4444h	19.2.3/796
108	Peripheral Access Control Register (PBRIDGE_PACRC)	32	R/W	4444_4444h	19.2.3/796
10C	Peripheral Access Control Register (PBRIDGE_PACRD)	32	R/W	4444_4444h	19.2.3/796
110	Peripheral Access Control Register (PBRIDGE_PACRE)	32	R/W	4444_4444h	19.2.3/796
114	Peripheral Access Control Register (PBRIDGE_PACRF)	32	R/W	4444_4444h	19.2.3/796
118	Peripheral Access Control Register (PBRIDGE_PACRG)	32	R/W	4444_4444h	19.2.3/796
11C	Peripheral Access Control Register (PBRIDGE_PACRH)	32	R/W	4444_4444h	19.2.3/796
140	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRA)	32	R/W	4444_4444h	19.2.4/798
144	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRB)	32	R/W	4444_4444h	19.2.4/798
148	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRC)	32	R/W	4444_4444h	19.2.4/798
14C	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRD)	32	R/W	4444_4444h	19.2.4/798
150	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRE)	32	R/W	4444_4444h	19.2.4/798

Table continues on the next page...

PBRIDGE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
154	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRF)	32	R/W	4444_4444h	19.2.4/798
158	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRG)	32	R/W	4444_4444h	19.2.4/798
15C	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRH)	32	R/W	4444_4444h	19.2.4/798
160	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRI)	32	R/W	4444_4444h	19.2.4/798
164	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRJ)	32	R/W	4444_4444h	19.2.4/798
168	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRK)	32	R/W	4444_4444h	19.2.4/798
16C	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRL)	32	R/W	4444_4444h	19.2.4/798
170	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRM)	32	R/W	4444_4444h	19.2.4/798
174	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRN)	32	R/W	4444_4444h	19.2.4/798
178	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRO)	32	R/W	4444_4444h	19.2.4/798
17C	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRP)	32	R/W	4444_4444h	19.2.4/798
180	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRQ)	32	R/W	4444_4444h	19.2.4/798
184	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRR)	32	R/W	4444_4444h	19.2.4/798
188	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRS)	32	R/W	4444_4444h	19.2.4/798
18C	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRT)	32	R/W	4444_4444h	19.2.4/798
190	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRU)	32	R/W	4444_4444h	19.2.4/798
194	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRV)	32	R/W	4444_4444h	19.2.4/798
198	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRW)	32	R/W	4444_4444h	19.2.4/798
19C	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRX)	32	R/W	4444_4444h	19.2.4/798
1A0	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRY)	32	R/W	4444_4444h	19.2.4/798
1A4	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRZ)	32	R/W	4444_4444h	19.2.4/798
1A8	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRAA)	32	R/W	4444_4444h	19.2.4/798

Table continues on the next page...

PBRIDGE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1AC	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRAB)	32	R/W	4444_4444h	19.2.4/798
1B0	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRAC)	32	R/W	4444_4444h	19.2.4/798
1B4	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRAD)	32	R/W	4444_4444h	19.2.4/798
1B8	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRAE)	32	R/W	4444_4444h	19.2.4/798
1BC	Off-platform Peripheral Access Control Register (PBRIDGE_OPACRAF)	32	R/W	4444_4444h	19.2.4/798

19.2.1 Master Privilege Register A (PBRIDGE_MPRA)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. See Crossbar switch configuration for details about the master assignments to these registers.

The MPROT*n* field is defined as shown in Figure 19-1 and Table 19-1.

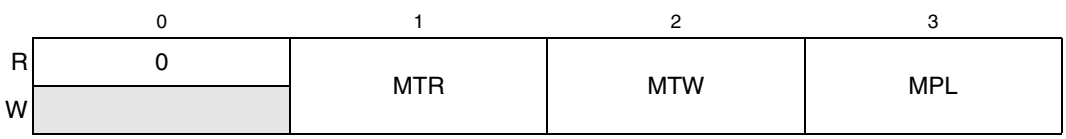


Figure 19-1. MPROT*n* fields

Table 19-1. MPROT*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 MTR	Master trusted for read Determines whether the master is trusted for read accesses. 0: This master is not trusted for read accesses. 1: This master is trusted for read accesses.
2 MTW	Master trusted for writes Determines whether the master is trusted for write accesses. 0: This master is not trusted for write accesses. 1: This master is trusted for write accesses.
3 MPL	Master privilege level Determines how the privilege level of the master is determined.

Table 19-1. MPROT*n* field descriptions

Field	Description
	0: Accesses from this master are forced to user mode. 1: Accesses from this master are not forced to user mode.

Address: F800_0000h base + 0h offset = F800_0000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MPROT0				MPROT1				MPROT2				MPROT3				MPROT4				MPROT5				MPROT6				MPROT7			
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

PBRIDGE_MPRA field descriptions

Field	Description
0–3 MPROT0	Access privilege level associated with bus master 0
4–7 MPROT1	Access privilege level associated with bus master 1
8–11 MPROT2	Access privilege level associated with bus master 2
12–15 MPROT3	Access privilege level associated with bus master 3
16–19 MPROT4	Access privilege level associated with bus master 4
20–23 MPROT5	Access privilege level associated with bus master 5
24–27 MPROT6	Access privilege level associated with bus master 6
28–31 MPROT7	Access privilege level associated with bus master 7

19.2.2 Master Privilege Register B (PBRIDGE_MPRB)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. See Crossbar switch configuration for details about the master assignments to these registers.

The MPROT*n* field is defined as shown in Figure 19-1 and Table 19-1.

Address: F800_0000h base + 4h offset = F800_0004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MPROT8				MPROT9				MPROT10				MPROT11				MPROT12				MPROT13				MPROT14				MPROT15			
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

PBRIDGE_MPRB field descriptions

Field	Description
0–3 MPROT8	Access privilege level associated with bus master 8
4–7 MPROT9	Access privilege level associated with bus master 9
8–11 MPROT10	Access privilege level associated with bus master 10
12–15 MPROT11	Access privilege level associated with bus master 11
16–19 MPROT12	Access privilege level associated with bus master 12
20–23 MPROT13	Access privilege level associated with bus master 13
24–27 MPROT14	Access privilege level associated with bus master 14
28–31 MPROT15	Access privilege level associated with bus master 15

19.2.3 Peripheral Access Control Register (PBRIDGE_PACR*n*)

Each of the on-platform peripherals has a four-bit PACR*n* field which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR*x* register.

The peripheral assignments to each PACR field are defined by the memory map slot to which the peripherals are assigned. See Table 5-2 for the assignments for your particular device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeroes, and writes are ignored.

Each PACR*n* field is defined as shown in Figure 19-2.

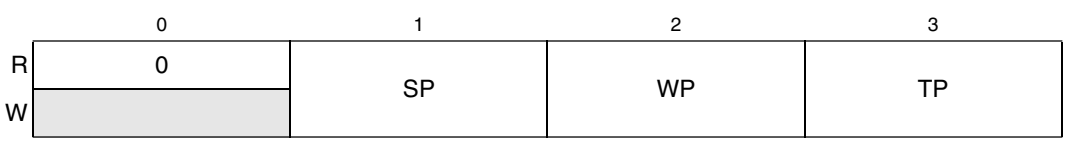


Figure 19-2. PACR*n* fields

The following table shows the top-level structure of the PACRs.

Table 19-2. PACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x0104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15

Table continues on the next page...

Table 19-2. PACR memory map (continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x010C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x0110	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x0118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x011C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Table 19-3. PACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0: This peripheral does not require supervisor privilege level for accesses. 1: This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0: This peripheral allows write accesses. 1: This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0: Accesses from an untrusted master are allowed. 1: Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

NOTE

The reset value of PACR0 is 0101b.

Address: F800_0000h base + 100h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PACRa				PACRb				PACRc				PACRd				PACRe				PACRf				PACRg				PACRh			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

PBRIDGE_PACR*n* field descriptions

Field	Description
0–3 PACRa	Access level associated with module a

Table continues on the next page...

PBRIDGE_PACR*n* field descriptions (continued)

Field	Description
4–7 PACRb	Access level associated with module b
8–11 PACRc	Access level associated with module c
12–15 PACRd	Access level associated with module d
16–19 PACRe	Access level associated with module e
20–23 PACRf	Access level associated with module f
24–27 PACRg	Access level associated with module g
28–31 PACRh	Access level associated with module h

19.2.4 Off-platform Peripheral Access Control Register (PBRIDGE_OPACR*n*)

Each of the off-platform peripherals has a four-bit OPACR*n* field which defines the access levels supported by the given module. Eight OPACRs are grouped together to form a 32-bit OPACR*x* register.

The peripheral assignments to each OPACR field are defined by the memory map slot to which the peripherals are assigned. See Table 5-2 for the assignments for your particular device. If a peripheral is absent, the corresponding OPACR is not implemented. Reads to the location return zeroes and writes are ignored.

The following table shows the top-level structure of the OPACR registers.

Table 19-4. OPACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0: This peripheral does not require supervisor privilege level for accesses. 1: This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0: This peripheral allows write accesses.

Table continues on the next page...

Table 19-4. OPACR*n* field descriptions (continued)

Field	Description
	1: This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0: Accesses from an untrusted master are allowed. 1: Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

Table 19-5. OPACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0140	OPACRA	OPACR 0	OPACR 1	OPACR 2	OPACR 3	OPACR 4	OPACR 5	OPACR 6	OPACR 7
0x0144	OPACRB	OPACR8	OPACR 9	OPACR 10	OPACR 11	OPACR 12	OPACR 13	OPACR 14	OPACR 15
0x0148	OPACRC	OPACR 16	OPACR 17	OPACR 18	OPACR 19	OPACR 20	OPACR 21	OPACR 22	OPACR 23
0x014C	OPACRD	OPACR 24	OPACR 25	OPACR 26	OPACR 27	OPACR 28	OPACR 29	OPACR 30	OPACR 31
0x0150	OPACRE	OPACR 32	OPACR 33	OPACR 34	OPACR 35	OPACR 36	OPACR 37	OPACR 38	OPACR 39
0x0154	OPACRF	OPACR 40	OPACR 41	OPACR 42	OPACR 43	OPACR 44	OPACR 45	OPACR 46	OPACR 47
0x0158	OPACRG	OPACR 48	OPACR 49	OPACR 50	OPACR 51	OPACR 52	OPACR 53	OPACR 54	OPACR 55
0x015C	OPACRH	OPACR 56	OPACR 57	OPACR 58	OPACR 59	OPACR 60	OPACR 61	OPACR 62	OPACR 63
0x0160	OPACRI	OPACR 64	OPACR 65	OPACR 66	OPACR 67	OPACR 68	OPACR 69	OPACR 70	OPACR 71
0x0164	OPACRJ	OPACR 72	OPACR 73	OPACR 74	OPACR 75	OPACR 76	OPACR 77	OPACR 78	OPACR 79
0x0168	OPACRK	OPACR 80	OPACR 81	OPACR 82	OPACR 83	OPACR 84	OPACR 85	OPACR 86	OPACR 87
0x016C	OPACRL	OPACR 88	OPACR 89	OPACR 90	OPACR 91	OPACR 92	OPACR 93	OPACR 94	OPACR 95
0x0170	OPACRM	OPACR 96	OPACR 97	OPACR 98	OPACR 99	OPACR 100	OPACR 101	OPACR 102	OPACR 103
0x0174	OPACRN	OPACR 104	OPACR 105	OPACR 106	OPACR 107	OPACR 108	OPACR 109	OPACR 110	OPACR 111

Table continues on the next page...

Table 19-5. OPACR memory map (continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0178	OPACRO	OPACR 112	OPACR 113	OPACR 114	OPACR 115	OPACR 116	OPACR 117	OPACR 118	OPACR 119
0x017C	OPACRP	OPACR 120	OPACR 121	OPACR 122	OPACR 123	OPACR 124	OPACR 125	OPACR 126	OPACR 127
0x0180	OPACRQ	OPACR 128	OPACR 129	OPACR 130	OPACR 131	OPACR 132	OPACR 133	OPACR 134	OPACR 135
0x0184	OPACRR	OPACR 136	OPACR 137	OPACR 138	OPACR 139	OPACR 140	OPACR 141	OPACR 142	OPACR 143
0x0188	OPACRS	OPACR 144	OPACR 145	OPACR 146	OPACR 147	OPACR 148	OPACR 149	OPACR 150	OPACR 151
0x018C	OPACRT	OPACR 152	OPACR 153	OPACR 154	OPACR 155	OPACR 156	OPACR 157	OPACR 158	OPACR 159
0x0190	OPACRU	OPACR 160	OPACR 161	OPACR 162	OPACR 163	OPACR 164	OPACR 165	OPACR 166	OPACR 167
0x0194	OPACRV	OPACR 168	OPACR 169	OPACR 170	OPACR 171	OPACR 172	OPACR 173	OPACR 174	OPACR 175
0x0198	OPACRW	OPACR 176	OPACR 177	OPACR 178	OPACR 179	OPACR 180	OPACR 181	OPACR 182	OPACR 183
0x019C	OPACRX	OPACR 184	OPACR 185	OPACR 186	OPACR 187	OPACR 188	OPACR 189	OPACR 190	OPACR 191
0x01A0	OPACRY	OPACR 192	OPACR 193	OPACR 194	OPACR 195	OPACR 196	OPACR 197	OPACR 198	OPACR 199
0x01A4	OPACRZ	OPACR 200	OPACR 201	OPACR 202	OPACR 203	OPACR 204	OPACR 205	OPACR 206	OPACR 207
0x01A8	OPACRAA	OPACR 208	OPACR 209	OPACR 210	OPACR 211	OPACR 212	OPACR 213	OPACR 214	OPACR 215
0x01AC	OPACRAB	OPACR 216	OPACR 217	OPACR 218	OPACR 219	OPACR 220	OPACR 221	OPACR 222	OPACR 223
0x01B0	OPACRAC	OPACR 224	OPACR 225	OPACR 226	OPACR 227	OPACR 228	OPACR 229	OPACR 230	OPACR 231
0x01B4	OPACRAD	OPACR 232	OPACR 233	OPACR 234	OPACR 235	OPACR 236	OPACR 237	OPACR 238	OPACR 239
0x01B8	OPACRAE	OPACR 240	OPACR 241	OPACR 242	OPACR 243	OPACR 244	OPACR 245	OPACR 246	OPACR 247
0x01BC	OPACRAF	OPACR 248	OPACR 249	OPACR 250	OPACR 251	OPACR 252	OPACR 253	OPACR 254	OPACR 255

The OPACR*n* field is defined as in the following figure.

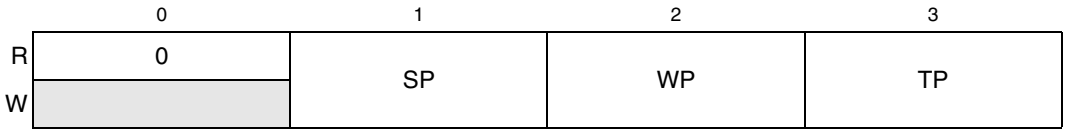


Figure 19-3. OPACR*n* fields

Address: F800_0000h base + 140h offset + (4d × i), where i=0d to 31d

Bit	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R W	OPACRa	OPACRb	OPACRc	OPACRd	OPACRe	OPACRf	OPACRg	OPACRh
Reset	0100	0100	0100	0100	0100	0100	0100	0100

PBRIDGE_OPACR*n* field descriptions

Field	Description
0–3 OPACRa	Access level associated with module a
4–7 OPACRb	Access level associated with module b
8–11 OPACRc	Access level associated with module c
12–15 OPACRd	Access level associated with module d
16–19 OPACRe	Access level associated with module e
20–23 OPACRf	Access level associated with module f
24–27 OPACRg	Access level associated with module g
28–31 OPACRh	Access level associated with module h

19.3 Functional description

The peripheral bridge serves as an interface between the crossbar switch and the slave peripheral bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

Accesses which fall within the address space of the peripheral bridge are decoded to provide individual module selects for peripheral devices on the slave bus interface.

19.3.1 Access support

Aligned and misaligned 32-bit and 16-bit accesses, as well as byte accesses, are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only. All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted which is larger (in size) than the targeted port, an error response is generated without a peripheral access.

Chapter 20
System Memory Protection Unit (SMPU)

20.1 Overview

The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using pre-programmed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

20.2 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, see Access evaluation macro.

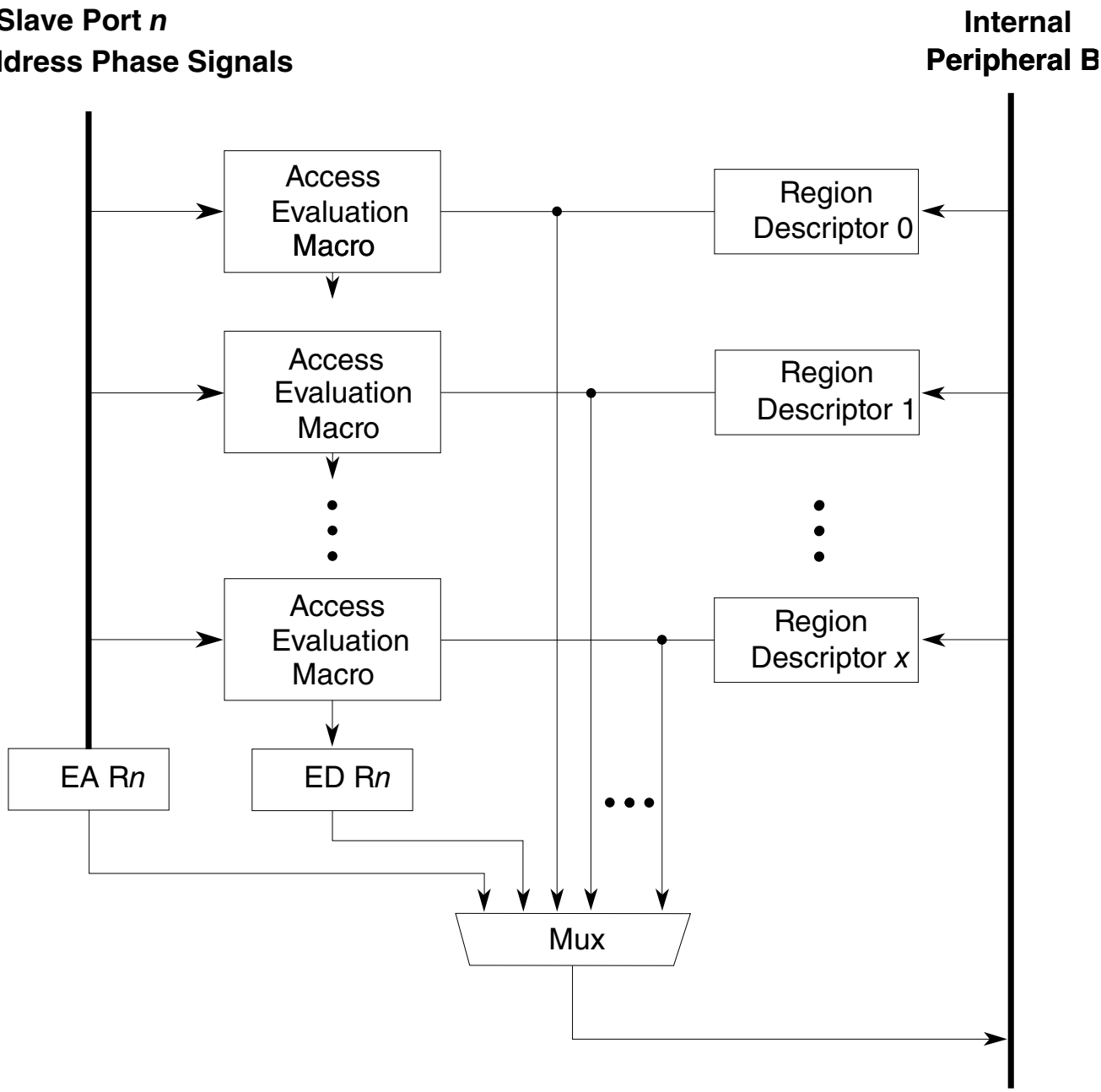


Figure 20-1. SMPU block diagram

20.3 Features

The SMPU feature set includes:

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24.)

    - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
        - Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
    - Read/write access control permissions defined in region descriptor
    - Cache-inhibit attribute indicator per region descriptor
    - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
    - Priority given to granting permission over denying access for overlapping region descriptors
- Supports as many as eight crossbar slave ports
- Detects access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Aborted core accesses generate instruction storage (ISI) or data storage (DSI) interrupts (see the documentation related to the core for details.)
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information

20.4 Memory map and register definition

The programming model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

The programming model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (a write to a read-only register, or a read of a write-only register) generate an error termination.

The programming model can be accessed only in supervisor mode.

NOTE

See the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to eight region descriptors.

SMPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control/Error Status Register 0 (SMPU_CESR0)	32	R/W	0000_8002h	20.4.1/809
4	Control/Error Status Register 1 (SMPU_CESR1)	32	R	See section	20.4.2/810
100	Error Address Register, Bus Master n (SMPU_EAR0)	32	R	0000_0000h	20.4.3/811
104	Error Detail Register, Bus Master n (SMPU_EDR0)	32	R	0000_0080h	20.4.4/811
108	Error Address Register, Bus Master n (SMPU_EAR1)	32	R	0000_0000h	20.4.3/811
10C	Error Detail Register, Bus Master n (SMPU_EDR1)	32	R	0000_0080h	20.4.4/811
110	Error Address Register, Bus Master n (SMPU_EAR2)	32	R	0000_0000h	20.4.3/811
114	Error Detail Register, Bus Master n (SMPU_EDR2)	32	R	0000_0080h	20.4.4/811
118	Error Address Register, Bus Master n (SMPU_EAR3)	32	R	0000_0000h	20.4.3/811
11C	Error Detail Register, Bus Master n (SMPU_EDR3)	32	R	0000_0080h	20.4.4/811
120	Error Address Register, Bus Master n (SMPU_EAR4)	32	R	0000_0000h	20.4.3/811
124	Error Detail Register, Bus Master n (SMPU_EDR4)	32	R	0000_0080h	20.4.4/811
128	Error Address Register, Bus Master n (SMPU_EAR5)	32	R	0000_0000h	20.4.3/811
12C	Error Detail Register, Bus Master n (SMPU_EDR5)	32	R	0000_0080h	20.4.4/811
130	Error Address Register, Bus Master n (SMPU_EAR6)	32	R	0000_0000h	20.4.3/811
134	Error Detail Register, Bus Master n (SMPU_EDR6)	32	R	0000_0080h	20.4.4/811
138	Error Address Register, Bus Master n (SMPU_EAR7)	32	R	0000_0000h	20.4.3/811
13C	Error Detail Register, Bus Master n (SMPU_EDR7)	32	R	0000_0080h	20.4.4/811
140	Error Address Register, Bus Master n (SMPU_EAR8)	32	R	0000_0000h	20.4.3/811
144	Error Detail Register, Bus Master n (SMPU_EDR8)	32	R	0000_0080h	20.4.4/811
148	Error Address Register, Bus Master n (SMPU_EAR9)	32	R	0000_0000h	20.4.3/811
14C	Error Detail Register, Bus Master n (SMPU_EDR9)	32	R	0000_0080h	20.4.4/811
150	Error Address Register, Bus Master n (SMPU_EAR10)	32	R	0000_0000h	20.4.3/811
154	Error Detail Register, Bus Master n (SMPU_EDR10)	32	R	0000_0080h	20.4.4/811
158	Error Address Register, Bus Master n (SMPU_EAR11)	32	R	0000_0000h	20.4.3/811
15C	Error Detail Register, Bus Master n (SMPU_EDR11)	32	R	0000_0080h	20.4.4/811
160	Error Address Register, Bus Master n (SMPU_EAR12)	32	R	0000_0000h	20.4.3/811
164	Error Detail Register, Bus Master n (SMPU_EDR12)	32	R	0000_0080h	20.4.4/811
168	Error Address Register, Bus Master n (SMPU_EAR13)	32	R	0000_0000h	20.4.3/811
16C	Error Detail Register, Bus Master n (SMPU_EDR13)	32	R	0000_0080h	20.4.4/811
170	Error Address Register, Bus Master n (SMPU_EAR14)	32	R	0000_0000h	20.4.3/811
174	Error Detail Register, Bus Master n (SMPU_EDR14)	32	R	0000_0080h	20.4.4/811
178	Error Address Register, Bus Master n (SMPU_EAR15)	32	R	0000_0000h	20.4.3/811

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17C	Error Detail Register, Bus Master n (SMPU_EDR15)	32	R	0000_0080h	20.4.4/811
400	Region Descriptor n, Word 0 (SMPU_RGD0_WORD0)	32	R/W	0000_0000h	20.4.5/812
404	Region Descriptor n, Word 1 (SMPU_RGD0_WORD1)	32	R/W	0000_0000h	20.4.6/813
408	Region Descriptor n, Word 2 Format 0 (SMPU_RGD0_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
40C	Region Descriptor n, Word 3 (SMPU_RGD0_WORD3)	32	R/W	0000_0000h	20.4.8/815
410	Region Descriptor n, Word 0 (SMPU_RGD1_WORD0)	32	R/W	0000_0000h	20.4.5/812
414	Region Descriptor n, Word 1 (SMPU_RGD1_WORD1)	32	R/W	0000_0000h	20.4.6/813
418	Region Descriptor n, Word 2 Format 0 (SMPU_RGD1_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
41C	Region Descriptor n, Word 3 (SMPU_RGD1_WORD3)	32	R/W	0000_0000h	20.4.8/815
420	Region Descriptor n, Word 0 (SMPU_RGD2_WORD0)	32	R/W	0000_0000h	20.4.5/812
424	Region Descriptor n, Word 1 (SMPU_RGD2_WORD1)	32	R/W	0000_0000h	20.4.6/813
428	Region Descriptor n, Word 2 Format 0 (SMPU_RGD2_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
42C	Region Descriptor n, Word 3 (SMPU_RGD2_WORD3)	32	R/W	0000_0000h	20.4.8/815
430	Region Descriptor n, Word 0 (SMPU_RGD3_WORD0)	32	R/W	0000_0000h	20.4.5/812
434	Region Descriptor n, Word 1 (SMPU_RGD3_WORD1)	32	R/W	0000_0000h	20.4.6/813
438	Region Descriptor n, Word 2 Format 0 (SMPU_RGD3_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
43C	Region Descriptor n, Word 3 (SMPU_RGD3_WORD3)	32	R/W	0000_0000h	20.4.8/815
440	Region Descriptor n, Word 0 (SMPU_RGD4_WORD0)	32	R/W	0000_0000h	20.4.5/812
444	Region Descriptor n, Word 1 (SMPU_RGD4_WORD1)	32	R/W	0000_0000h	20.4.6/813
448	Region Descriptor n, Word 2 Format 0 (SMPU_RGD4_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
44C	Region Descriptor n, Word 3 (SMPU_RGD4_WORD3)	32	R/W	0000_0000h	20.4.8/815
450	Region Descriptor n, Word 0 (SMPU_RGD5_WORD0)	32	R/W	0000_0000h	20.4.5/812
454	Region Descriptor n, Word 1 (SMPU_RGD5_WORD1)	32	R/W	0000_0000h	20.4.6/813
458	Region Descriptor n, Word 2 Format 0 (SMPU_RGD5_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
45C	Region Descriptor n, Word 3 (SMPU_RGD5_WORD3)	32	R/W	0000_0000h	20.4.8/815
460	Region Descriptor n, Word 0 (SMPU_RGD6_WORD0)	32	R/W	0000_0000h	20.4.5/812
464	Region Descriptor n, Word 1 (SMPU_RGD6_WORD1)	32	R/W	0000_0000h	20.4.6/813
468	Region Descriptor n, Word 2 Format 0 (SMPU_RGD6_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
46C	Region Descriptor n, Word 3 (SMPU_RGD6_WORD3)	32	R/W	0000_0000h	20.4.8/815
470	Region Descriptor n, Word 0 (SMPU_RGD7_WORD0)	32	R/W	0000_0000h	20.4.5/812
474	Region Descriptor n, Word 1 (SMPU_RGD7_WORD1)	32	R/W	0000_0000h	20.4.6/813
478	Region Descriptor n, Word 2 Format 0 (SMPU_RGD7_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
47C	Region Descriptor n, Word 3 (SMPU_RGD7_WORD3)	32	R/W	0000_0000h	20.4.8/815

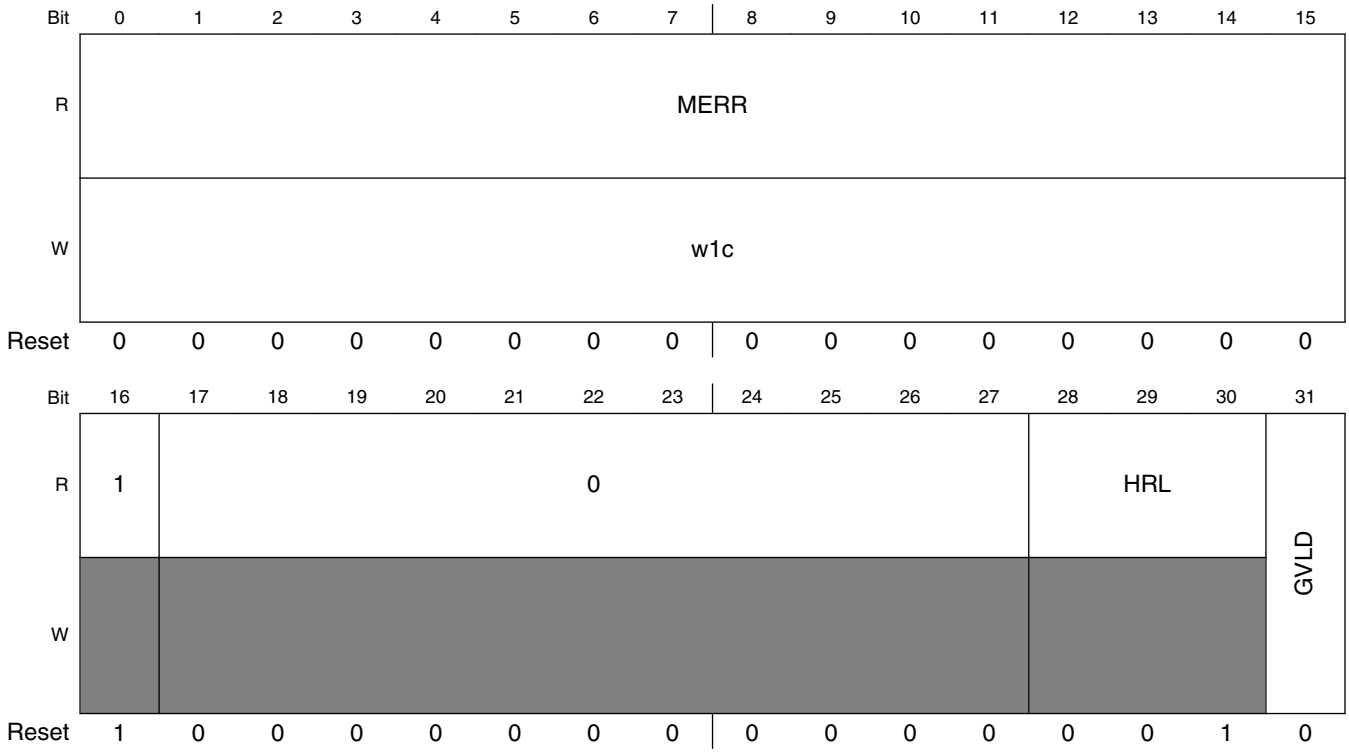
Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
480	Region Descriptor n, Word 0 (SMPU_RGD8_WORD0)	32	R/W	0000_0000h	20.4.5/812
484	Region Descriptor n, Word 1 (SMPU_RGD8_WORD1)	32	R/W	0000_0000h	20.4.6/813
488	Region Descriptor n, Word 2 Format 0 (SMPU_RGD8_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
48C	Region Descriptor n, Word 3 (SMPU_RGD8_WORD3)	32	R/W	0000_0000h	20.4.8/815
490	Region Descriptor n, Word 0 (SMPU_RGD9_WORD0)	32	R/W	0000_0000h	20.4.5/812
494	Region Descriptor n, Word 1 (SMPU_RGD9_WORD1)	32	R/W	0000_0000h	20.4.6/813
498	Region Descriptor n, Word 2 Format 0 (SMPU_RGD9_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
49C	Region Descriptor n, Word 3 (SMPU_RGD9_WORD3)	32	R/W	0000_0000h	20.4.8/815
4A0	Region Descriptor n, Word 0 (SMPU_RGD10_WORD0)	32	R/W	0000_0000h	20.4.5/812
4A4	Region Descriptor n, Word 1 (SMPU_RGD10_WORD1)	32	R/W	0000_0000h	20.4.6/813
4A8	Region Descriptor n, Word 2 Format 0 (SMPU_RGD10_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
4AC	Region Descriptor n, Word 3 (SMPU_RGD10_WORD3)	32	R/W	0000_0000h	20.4.8/815
4B0	Region Descriptor n, Word 0 (SMPU_RGD11_WORD0)	32	R/W	0000_0000h	20.4.5/812
4B4	Region Descriptor n, Word 1 (SMPU_RGD11_WORD1)	32	R/W	0000_0000h	20.4.6/813
4B8	Region Descriptor n, Word 2 Format 0 (SMPU_RGD11_WORD2_FMT0)	32	R/W	0000_0000h	20.4.7/813
4BC	Region Descriptor n, Word 3 (SMPU_RGD11_WORD3)	32	R/W	0000_0000h	20.4.8/815

20.4.1 Control/Error Status Register 0 (SMPU_CESR0)

Address: FC01_0000h base + 0h offset = FC01_0000h



SMPU_CESR0 field descriptions

Field	Description
0–15 MERR	Master n error, where the bus master number matches the bit number Indicates a captured error in EARn and EDRn. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing '1' to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master n. 1 An error has occurred for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled

20.4.2 Control/Error Status Register 1 (SMPU_CESR1)

Address: FC01_0000h base + 4h offset = FC01_0004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0											NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*

* Notes:
- NRGD field: NRGD[3] and NRGD[2] reset to 0. The reset value of the other bits is undefined.

SMPU_CESR1 field descriptions

Field	Description
0–15 MEOVR	Master n error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master n has occurred before the previous error was processed. The details of the first error are recorded in the EARn and EDRn registers, and no information on subsequent errors is recorded until the associated CESR0[MERR] flag is cleared. 0 No error overrun condition has been detected for bus master n. 1 An error overrun condition has been detected for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

20.4.3 Error Address Register, Bus Master n (SMPU_EAR*n*)

When the SMPU detects an access error on bus master n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Additional information about the faulting access is captured in the corresponding EDRn at the same time.

NOTE

The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the original error is processed.

Address: FC01_0000h base + 100h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_EAR*n* field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error

20.4.4 Error Detail Register, Bus Master n (SMPU_EDR*n*)

When the SMPU detects an access error associated with bus master n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Information on the faulting address is captured in the corresponding EARn register at the same time.

NOTE

The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the original error is processed.

Address: FC01_0000h base + 104h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EACD								1	EATTR		ERW	EMN			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

SMPU_EDR*n* field descriptions

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDRn contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
25–26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28–31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

20.4.5 Region Descriptor n, Word 0 (SMPU_RGD*n*_WORD0)

Address: FC01_0000h base + 400h offset + (16d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD0 field descriptions

Field	Description
0–31 SRTADDR	Start address Defines the byte start address of the memory region.

20.4.6 Region Descriptor n, Word 1 (SMPU_RGD*n*_WORD1)

Address: FC01_0000h base + 404h offset + (16d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD1 field descriptions

Field	Description
0–31 ENDADDR	End address Defines the byte end address of the memory region. **NOTE:** The SMPU does not verify that ENDADDR >= SRTADDR.

20.4.7 Region Descriptor n, Word 2 Format 0 (SMPU_RGD*n*_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3[FMT] field.

NOTE

RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags per logical bus master:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: FC01_0000h base + 408h offset + (16d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD2_FMT0 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions
26–27 M13P	Bus master 13 permissions

Table continues on the next page...

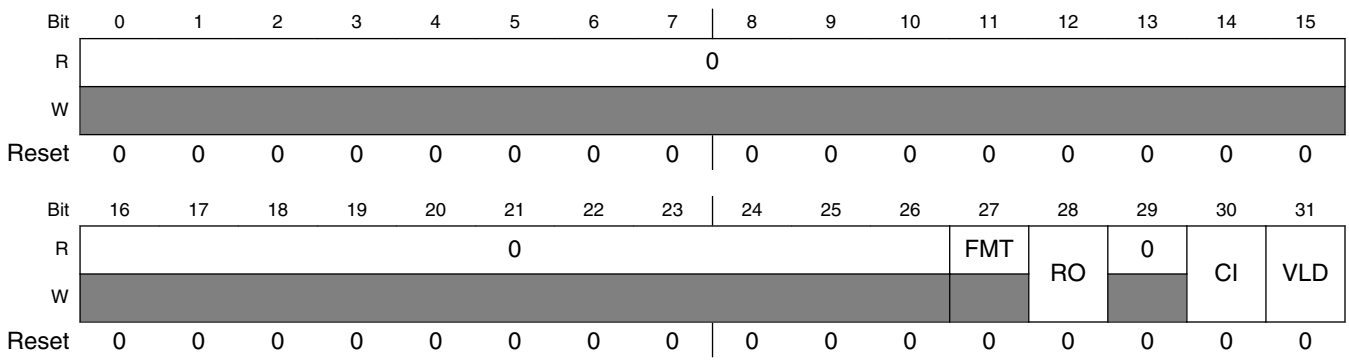
SMPU_RGD*n*_WORD2_FMT0 field descriptions (continued)

Field	Description
28–29 M14P	Bus master 14 permissions
30–31 M15P	Bus master 15 permissions

20.4.8 Region Descriptor n, Word 3 (SMPU_RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), plus other configuration bits.

Address: FC01_0000h base + 40Ch offset + (16d × i), where i=0d to 11d



SMPU_RGD*n*_WORD3 field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FMT	Region Descriptor Format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. **NOTE:** A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies. 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **NOTE:** Setting RO in an RGD locks all four words of the RGD until a system reset; the valid bit of the RGD and the global valid bit have no effect.

Table continues on the next page...

SMPU_RGD*n*_WORD3 field descriptions (continued)

Field	Description
	0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **NOTE:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0-2 clears this bit. 0 Region descriptor is invalid 1 Region descriptor is valid

20.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

20.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in Figure 20-2, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*), then performs two major functions: region hit determination and detection of an access protection violation. Figure 20-2 shows a functional block diagram.

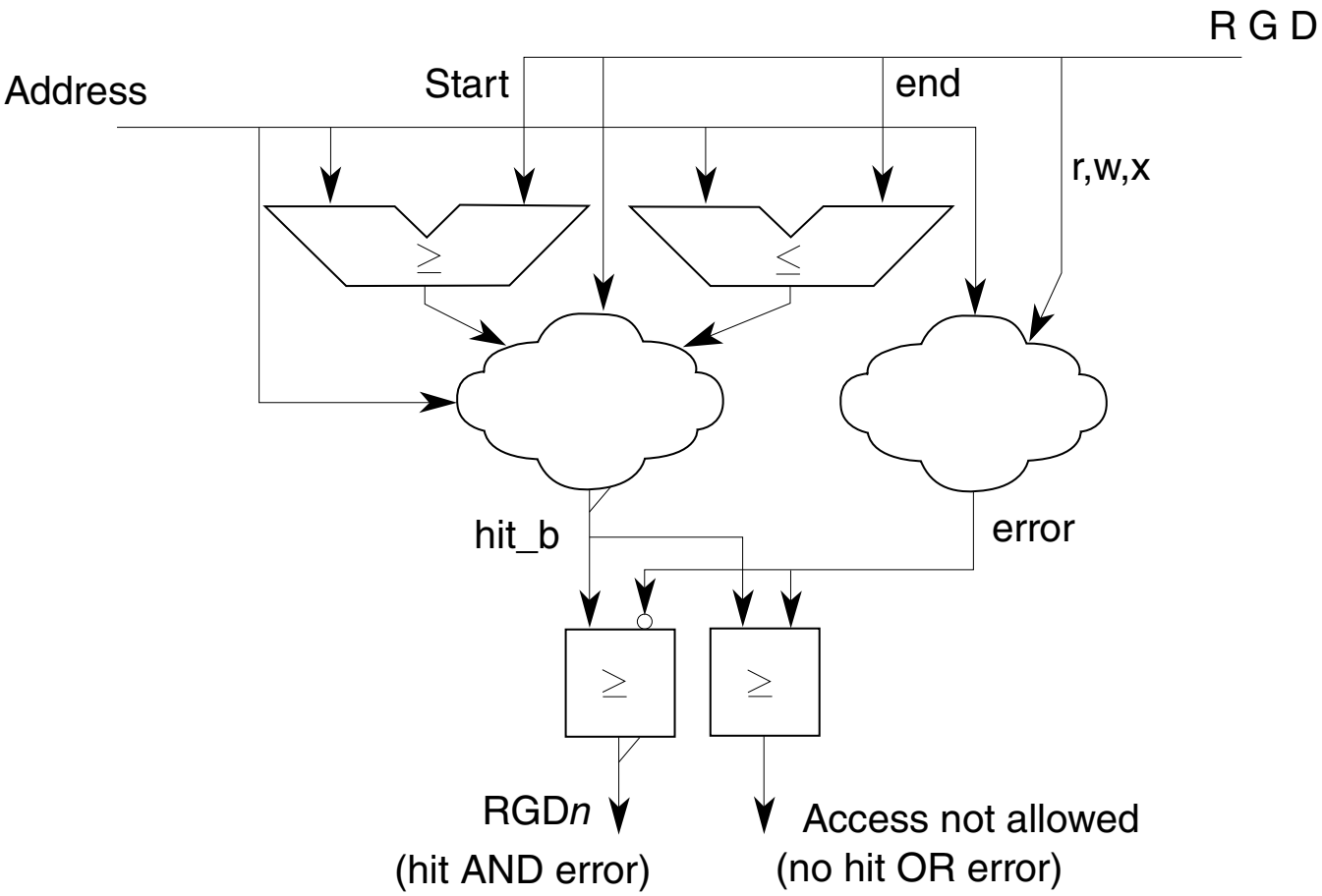


Figure 20-2. SMPU access evaluation macro

20.5.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit =          ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
                     (addr[0:31] <= RGDn_Word1[ENDADDR])) &
                     RGDn_Word3[VLD]
```

where addr is the current reference address, RGDn_Word0[SRTADDR] and RGDn_Word1[ENDADDR] are the start and end addresses, and RGDn_Word3[VLD] is the valid bit.

Note

The SMPU does not verify that ENDADDR ≥ SRTADDR.

Note

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

20.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in Table 20-1.

Table 20-1. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

20.5.2 Putting it all together and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Application information.

20.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGDn_Word3[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD] = 0). Next, load the appropriate region descriptors (RGD*n*), including the setting of the RGDn_Word3[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

20.7 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region—Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Modifying a region descriptor—Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Removing a region descriptor—Clearing RGD*n*_Word3[VLD] deletes an existing region descriptor.
- Accessing the SMPU—Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error—The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus

master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0[MERR] signals. The error registers contain the captured fault data.

- Overlapping region descriptors—Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters: the two processors (CP0, CP1) and two DMA engines (DMA1, a traditional data movement engine transferring data between RAM and peripherals, and DMA2, a second engine transferring data to/from the RAM only). Consider the following region descriptor assignments:

Table 20-2. Overlapping region descriptor example

Region description	RGDn	CP0	CP1	DMA1	DMA2	System memorymap space
CP0 code	0	rwx	r--	—	—	Flash
CP1 code	1	r--	rwx	—	—	
CP0 data & stack	2	rw-	—	—	—	RAM
CP0 → CP1 shared data	2 3	r--	r--	—	—	
CP1 → CP0 shared data	4					
CP1 data & stack	4	—	rw-	—	—	
Shared DMA data	5	rw-	rw-	rw	rw	
SMPU	6	rw-	rw-	—	—	Peripheral space
Peripherals	7	rw-	rw-	rw	—	

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map (flash, RAM, and peripheral space). Each region indicates the specific permissions for each of the four bus masters, and this definition provides an assigned set of shared, private, and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1, and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1

to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the peripheral bus is partitioned into two regions: one containing the SMPU's programming model accessible only to the two processor cores and the remaining peripheral region accessible to both processors and the traditional DMA1 master.

This simple example is intended to show one possible application of the capabilities of the SMPU in a typical system.

Chapter 21
Intelligent AHB Gasket (IAHBG)

21.1 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 21-1 lists the available operation modes.

Table 21-1. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

21.2 Timing modes

21.2.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. The pending read can either be a single or burst transaction.

21.2.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is not shown because it is very similar to the 2:1 mode. The pending read can either be a single or burst transaction.

21.2.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master. The 1:4 mode is not shown because it is very similar to the 1:2 mode. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write..

Chapter 22
Semaphores2 (SEMA42)

22.1 Introduction

The peripheral platform shell includes a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single write access. The hardware semaphore module is an architecture-neutral solution that provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

The capabilities provided by the hardware semaphore module are currently utilized in Freescale's dual-core AUTOSAR® implementations, and its use is expected to continue in future multi-core ports. The result is a second-generation module implementation known as Semaphores2 or SEMA42.

NOTE

What is AUTOSAR? From Wikipedia: AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers.

22.1.1 Multi-core programming 101: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism that is then used by system software to control access to shared data structures, shared hardware resources, and so on. These gating mechanisms are used by the software to serialize (and synchronize) accesses to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and processors.

Consider the following description of a typical use-case. Processor X enters a section of code where shared data values are to be updated. It must first acquire a semaphore; this can be considered to be locking (or closing) a software gate. Once the gate has been locked, a properly architected software system does not allow other processes (or processors) to execute the same code segment or modify the shared data structure protected by the gate; in other words, other processes/processors are locked out. Many software implementations include a *spin-wait loop* within the lock function until the locking of the gate is accomplished. Once the lock has been obtained, processor X continues execution and updates the data values protected by the particular lock. Once the updates are complete, processor X unlocks (or opens) the software gate, allowing other processes/processors access to the updated data values.

There are three important rules that must be followed for a correctly implemented system solution:

1. All writes to shared data values or shared hardware resources must be protected by a gate variable.
2. Once a processor locks a gate, accesses to the shared data or resources by other processes/processors must be blocked. This is enforced by software conventions.
3. The processor that locks a particular gate is the only processor that can open (unlock) that gate.

Information in the hardware gate identifying the locking processor can be extremely useful for system-level debugging.

The Hennessy/Patterson text on computer architecture offers the following description of software gating.

"One of the major requirements of a shared-memory architecture multiprocessor is being able to coordinate processes that are working on a common task. Typically, a programmer will use *lock variables* to synchronize the processes.

The difficulty for the architect of a multiprocessor is to provide a mechanism to decide which processor gets the lock and to provide the operation that locks a variable. Arbitration is easy for shared-bus multiprocessors, since the bus is the only path to memory. The processor that gets the bus locks out all the other processors from memory. If the CPU and bus provide an atomic swap operation, programmers can create locks with the proper semantics. The adjective *atomic* is key, for it means that a processor can both read a location **and** set it to the locked value in the same bus operation, preventing any other processor from reading or writing memory." [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*]

The classic text continues with a description of the steps required to lock/unlock a variable using an *atomic swap instruction*.

"Assume that 0 means unlocked and 1 means locked. A processor first reads the lock variable to test its state. A processor keeps reading and testing until the value indicates that the lock is unlocked. The processor then races against all other processes that were similarly "spin waiting" to see who can lock the variable first. All processes use a swap instruction that reads the old value and stores a 1 into the lock variable. The single winner will see the 0, and the losers will see a 1 that was placed there by the winner. (The losers will continue to set the variable to the locked value, but this does not matter.) The winning processor executes the code after the lock and then stores a 0 into the lock when it exits, starting the race all over again. Testing the old value and then setting to a new value is why the atomic swap instruction is called *test and set* in some instruction sets." [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*]

Next, consider the features, programming model, and functional operation of the hardware semaphore module.

22.1.2 Features

The Semaphores module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Module definition supporting 16 hardware-enforced gates in a multi-processor configuration, where up to 15 processors can be supported; cp*X* is meant to represent core processor *X*
  - Gates appear as an *n*-entry byte-size array with read and write accesses (*n* = 16, 32, 64).
    - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful.
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
    - 16-state implementation

      if gate = 0x0, then state = unlocked

if gate = 0x1, then state = locked by processor (master) 0
if gate = 0x2, then state = locked by processor (master) 1
…
if gate = 0xF, then state = locked by processor (master) 14

- Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
- Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.

- Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.

- Memory-mapped IPS slave peripheral platform module
  - Interface to the IPS bus for programming-model accesses

A simplified block diagram of the Semaphores module is shown in the following figure. In the diagram, the register blocks named gate0, gate1,…, gate (n-2), gate (n-1) include the finite state machines implementing the semaphore gates.

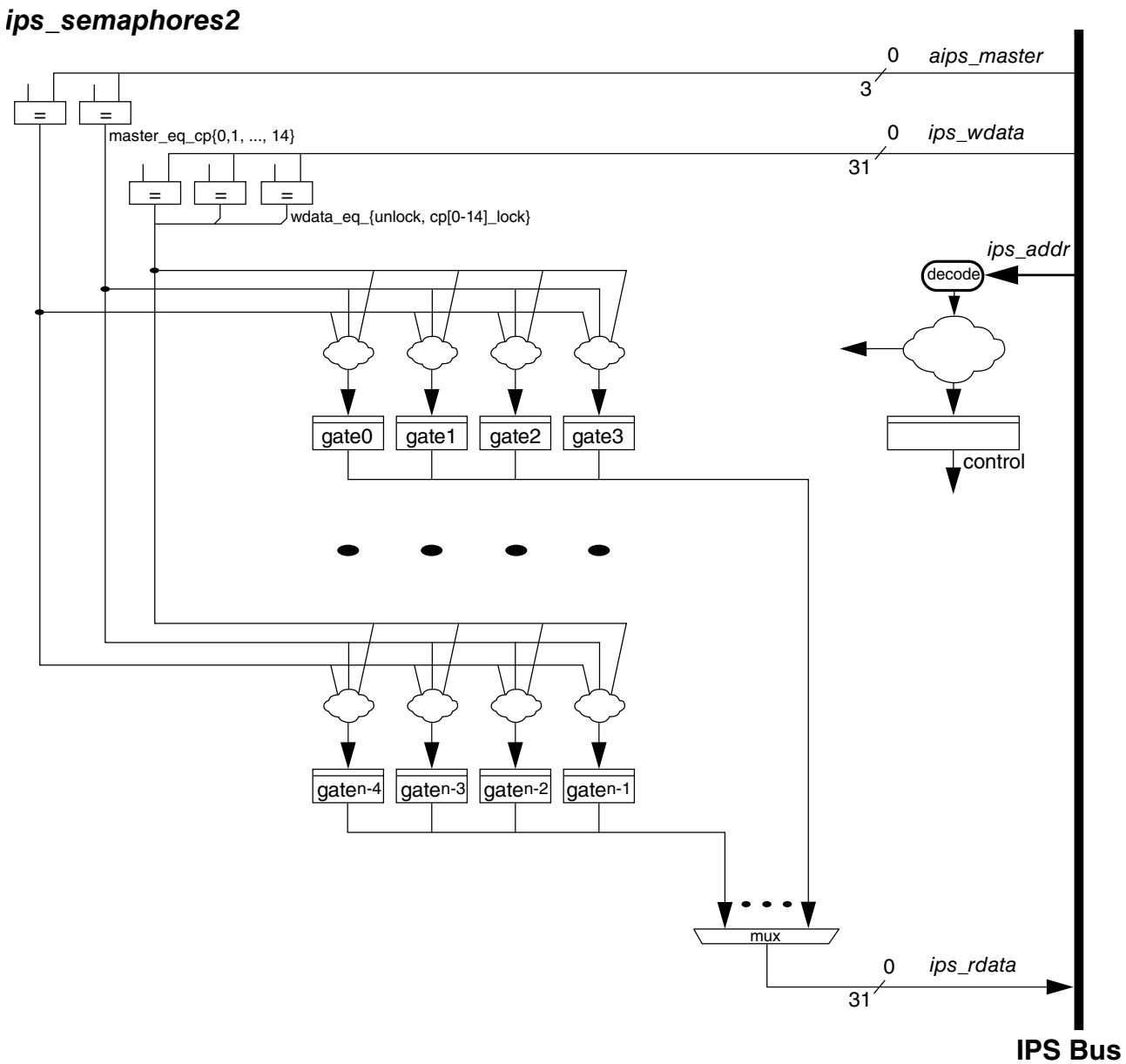


Figure 22-1. Semaphores2 block diagram

22.2 Memory map/register definition

Only Supervisor Mode accesses are allowed on these registers. User accesses generate an error termination.

SEMA42 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Gate Register (SEMA42_GATE0)	8	R/W	00h	22.2.1/830
1	Gate Register (SEMA42_GATE1)	8	R/W	00h	22.2.1/830
2	Gate Register (SEMA42_GATE2)	8	R/W	00h	22.2.1/830
3	Gate Register (SEMA42_GATE3)	8	R/W	00h	22.2.1/830
4	Gate Register (SEMA42_GATE4)	8	R/W	00h	22.2.1/830
5	Gate Register (SEMA42_GATE5)	8	R/W	00h	22.2.1/830
6	Gate Register (SEMA42_GATE6)	8	R/W	00h	22.2.1/830
7	Gate Register (SEMA42_GATE7)	8	R/W	00h	22.2.1/830
8	Gate Register (SEMA42_GATE8)	8	R/W	00h	22.2.1/830
9	Gate Register (SEMA42_GATE9)	8	R/W	00h	22.2.1/830
A	Gate Register (SEMA42_GATE10)	8	R/W	00h	22.2.1/830
B	Gate Register (SEMA42_GATE11)	8	R/W	00h	22.2.1/830
C	Gate Register (SEMA42_GATE12)	8	R/W	00h	22.2.1/830
D	Gate Register (SEMA42_GATE13)	8	R/W	00h	22.2.1/830
E	Gate Register (SEMA42_GATE14)	8	R/W	00h	22.2.1/830
F	Gate Register (SEMA42_GATE15)	8	R/W	00h	22.2.1/830
40	Reset Gate Write (SEMA42_RSTGT_W)	16	R/W	0000h	22.2.2/831
40	Reset Gate Read (SEMA42_RSTGT_R)	16	R/W	0000h	22.2.3/832

22.2.1 Gate Register (SEMA42_GATE*n*)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are simply treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read	0				GTFSM			
Write								
Reset	0	0	0	0	0	0	0	0

SEMA42_GATE*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 GTFSM	Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug. The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor 0. 0010 The gate has been locked by processor 1. 0011 The gate has been locked by processor 2. 0100 The gate has been locked by processor 3. 0101 The gate has been locked by processor 4. 0110 The gate has been locked by processor 5. 0111 The gate has been locked by processor 6. 1000 The gate has been locked by processor 7. 1001 The gate has been locked by processor 8. 1010 The gate has been locked by processor 9. 1011 The gate has been locked by processor 10. 1100 The gate has been locked by processor 11. 1101 The gate has been locked by processor 12. 1110 The gate has been locked by processor 13. 1111 The gate has been locked by processor 14.

22.2.2 Reset Gate Write (SEMA42_RSTGT_W)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA42_RSTGT memory location. The most significant byte (SEMA42_RSTGT[RSTGDP]) must be 0xE2; the least significant byte is a "don't_care" for this reference.
2. The same processor then performs a second 16-bit write to the SEMA42_RSTGT location. For this write, the upper byte (SEMA42_RSTGT[RSTGDP]) is the logical

complement of the first data pattern (0x1D) and the lower byte (SEMA42_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or in the case where SEMA42_RSTGT[RSTGTN] = 64 or greater, all gates are to be cleared. If the same processor writes incorrect data on the second access or another processor performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.

3. Reads of the SEMA42_RSTGT location return information on the 2-bit state machine (SEMA42_RSTGT[RSTGSM]) that implements this function, the bus master performing the reset (SEMA42_RSTGT[RSTGMS], and the gate number(s) last cleared (SEMA42_RSTGT[RSTGTN]). Reads of the SEMA42_RSTGT register do not affect the secure reset finite state machine in any manner.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								RSTGTN							
Write	RSTGDP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA42_RSTGT_W field descriptions

Field	Description
0–7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes by the same processor to enable the gate reset mechanism. For the first write, RSTGDP = 0xE2 while the second write requires RSTGDP = 0x1D.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

22.2.3 Reset Gate Read (SEMA42_RSTGT_R)

See description for Reset Gate Write (SEMA42_RSTGT_W).

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		RSTGSM		RSTGMS				RSTGTN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA42_RSTGT_R field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 RSTGSM	Reset Gate Finite State Machine. Reads of the SEMA42_RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software cannot observe this state. 11 This state encoding is never used and therefore reserved.
4–7 RSTGMS	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

22.3 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA42_GATEn registers are provided as follows.

As described previously, each of the SEMA42_GATEn registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in Figure 22-2.

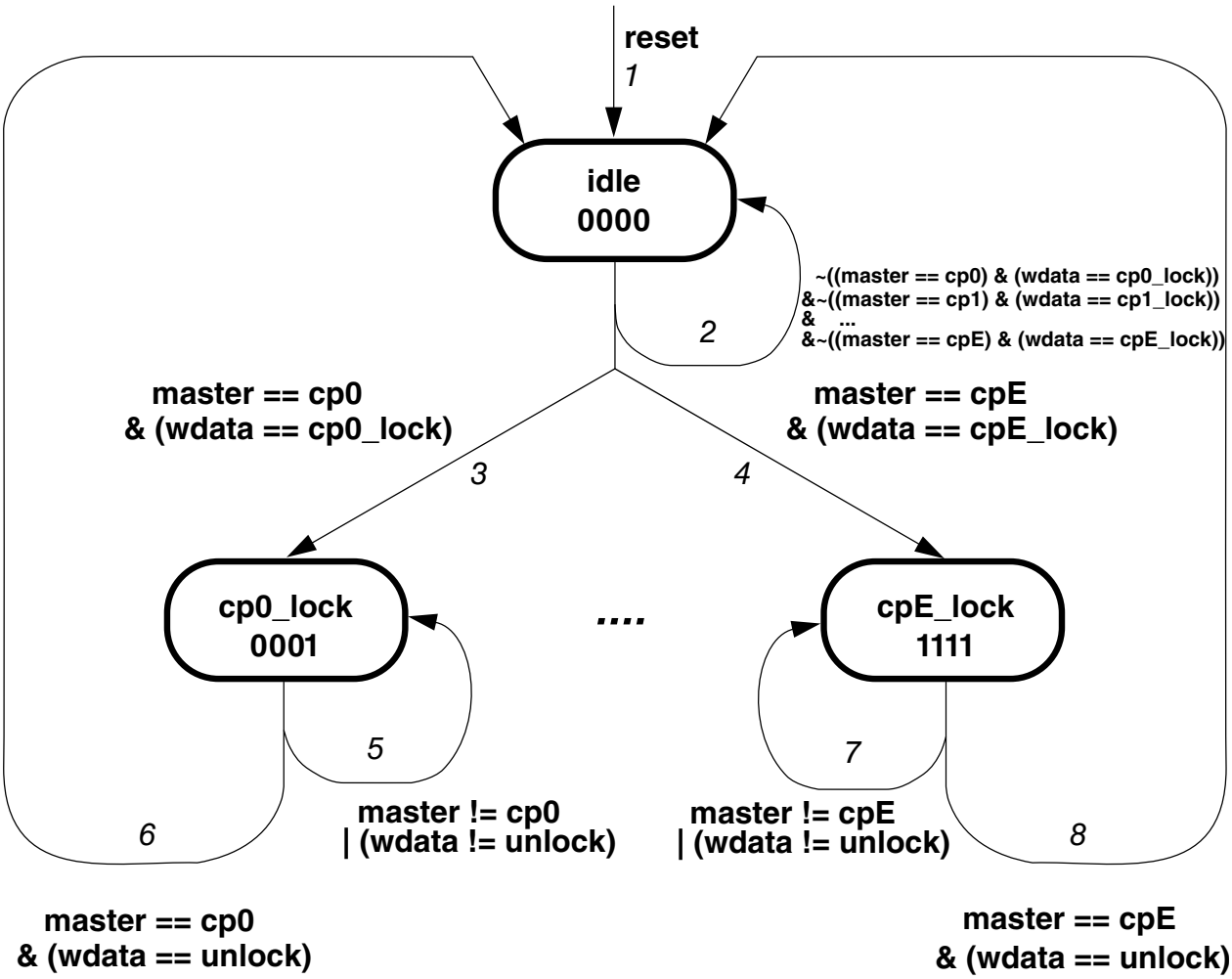


Figure 22-2. SEMA42_GATEn state machine

The bus master number is used to identify core processor *X* (cp*X*) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number (`hmaster[3:0]`) through the AIPS controller and drives the `aips_master[3:0]` signal to the Semaphores module as an IPS sideband signal.

The state transitions for SEMA42_GATEn are defined in Table 22-1.

Table 22-1. SEMA42_GATEn state transitions

Current State	Next State	Transition	Description
–	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.

Table continues on the next page...

Table 22-1. SEMA42_GATEn state transitions (continued)

Current State	Next State	Transition	Description
idle	cpE_lock	4	When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs.

The following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

Chapter 23
Interrupt Controller (INTC)

23.1 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

23.2 Block diagram

The following figure shows the block diagram of an INTC with four processors (PRC0...PRC3). The actual number of processors is described in the chip-specific INTC information. The structure of the INTC remains the same for each implemented processor.

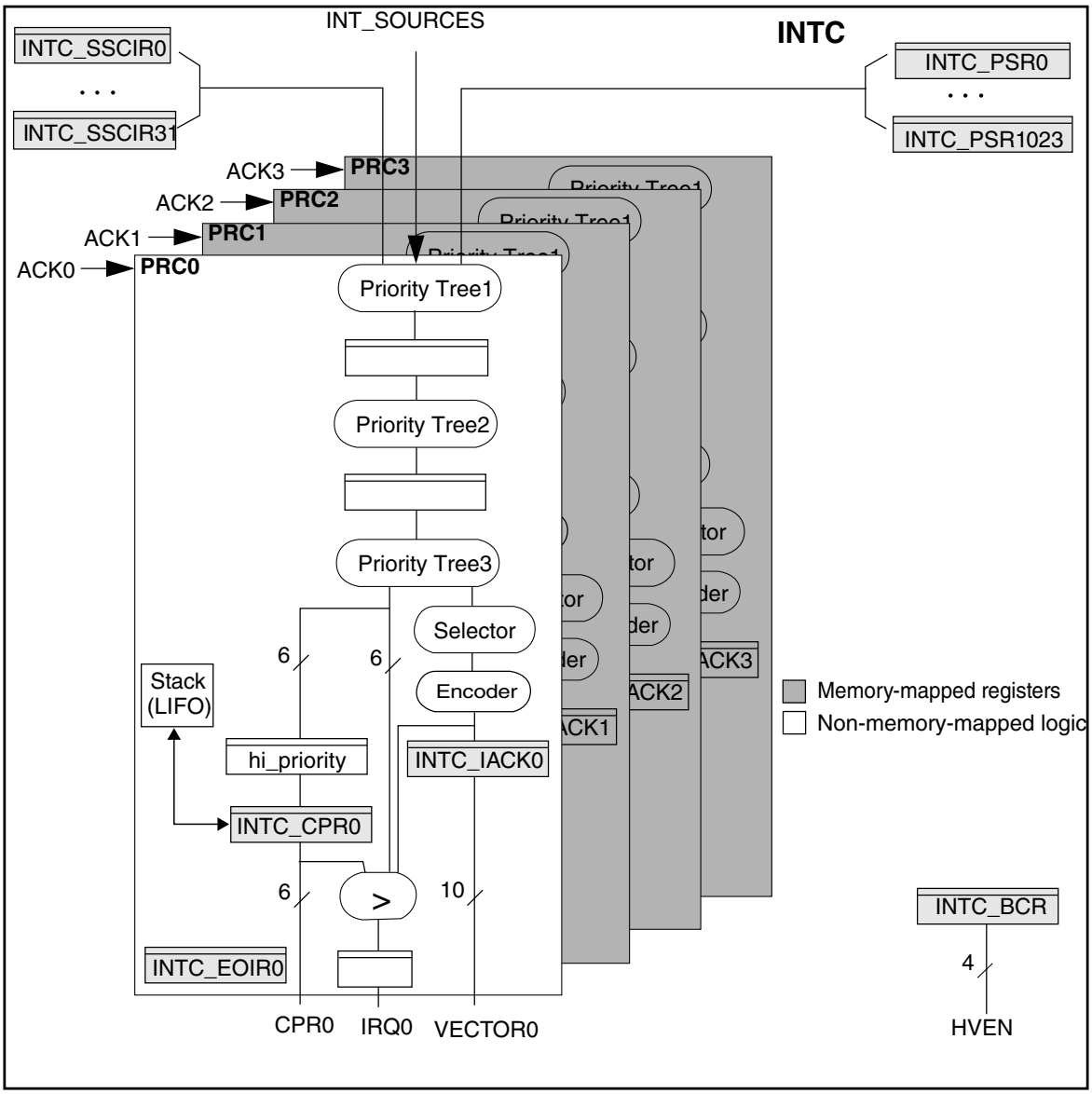


Figure 23-1. Block diagram for an INTC with four processors

23.3 Features

- Each peripheral interrupt source is software-steerable to any combination of interrupt request outputs to the following:
  - Processor 0
  - Processor 1
  - Processor 2
  - Processor 3

- 32 software-settable interrupt request sources
- 10-bit vector
  - Unique vector for each interrupt request source
  - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR with higher priority preempts ISRs or tasks with lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
  - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
  - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request

23.4 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

23.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN_PRCn bit in the INTC_BCR is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRC*n* bit is negated. The vector is read from the INTC_IACKRn registers. Reading the INTC_IACKR*n* negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in the INTC_CPRn onto the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

23.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVENn bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKRn, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*n* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*n* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR*n* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs is written at a time such that the PRI value in the associated INTC_CPR*n* register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*n* is updated with the new priority, and the associated LIFO is neither pushed or popped.

23.5 Memory map and register definition

A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

With the exception of the INTC_SSCIR *n* and INTC_PSR *n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

When writing to INTC_PSRs or INTC_SSCIRs, the following restrictions (illustrated in Figure 23-2) apply:

- For PSRs, write accesses must not span 16-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).
- For SSCIRs, write accesses must not span 8-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).

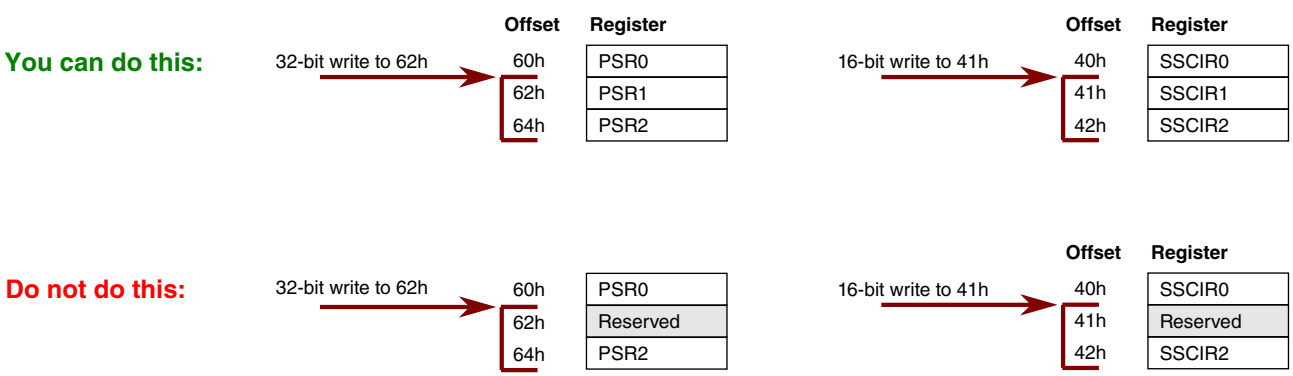


Figure 23-2. INTC_PSR*n* and INTC_SSCIR*n* write restrictions

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR *n*) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR *n*) does not affect the operation of the write.

INTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	INTC Block Configuration Register (INTC_BCR)	32	R/W	0000_0000h	23.5.1/891
4	INTC Master Protection Register (INTC_MPROT)	32	R/W	0000_0000h	23.5.2/893
10	INTC Current Priority Register for Processor 0 (INTC_CPR0)	32	R/W	0000_003Fh	23.5.3/894
14	INTC Current Priority Register for Processor 1 (INTC_CPR1)	32	R/W	0000_003Fh	23.5.4/895
18	INTC Current Priority Register for Processor 2 (INTC_CPR2)	32	R/W	0000_003Fh	23.5.5/896
1C	INTC Current Priority Register for Processor 3 (INTC_CPR3)	32	R/W	0000_003Fh	23.5.6/897
20	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	32	R/W	0000_0000h	23.5.7/898
24	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	32	R/W	0000_0000h	23.5.8/899
28	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)	32	R/W	0000_0000h	23.5.9/900
2C	INTC Interrupt Acknowledge Register for Processor 3 (INTC_IACKR3)	32	R/W	0000_0000h	23.5.10/ 901
30	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	32	W	0000_0000h	23.5.11/ 902
34	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	32	W	0000_0000h	23.5.12/ 902
38	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)	32	W	0000_0000h	23.5.13/ 903
3C	INTC End Of Interrupt Register for Processor 3 (INTC_EOIR3)	32	W	0000_0000h	23.5.14/ 904
40	INTC Software Set/Clear Interrupt Register (INTC_SSCIR0)	8	R/W	00h	23.5.15/ 905
41	INTC Software Set/Clear Interrupt Register (INTC_SSCIR1)	8	R/W	00h	23.5.15/ 905
42	INTC Software Set/Clear Interrupt Register (INTC_SSCIR2)	8	R/W	00h	23.5.15/ 905
43	INTC Software Set/Clear Interrupt Register (INTC_SSCIR3)	8	R/W	00h	23.5.15/ 905
44	INTC Software Set/Clear Interrupt Register (INTC_SSCIR4)	8	R/W	00h	23.5.15/ 905
45	INTC Software Set/Clear Interrupt Register (INTC_SSCIR5)	8	R/W	00h	23.5.15/ 905
46	INTC Software Set/Clear Interrupt Register (INTC_SSCIR6)	8	R/W	00h	23.5.15/ 905
47	INTC Software Set/Clear Interrupt Register (INTC_SSCIR7)	8	R/W	00h	23.5.15/ 905
48	INTC Software Set/Clear Interrupt Register (INTC_SSCIR8)	8	R/W	00h	23.5.15/ 905
49	INTC Software Set/Clear Interrupt Register (INTC_SSCIR9)	8	R/W	00h	23.5.15/ 905
4A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR10)	8	R/W	00h	23.5.15/ 905

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR11)	8	R/W	00h	23.5.15/ 905
4C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR12)	8	R/W	00h	23.5.15/ 905
4D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR13)	8	R/W	00h	23.5.15/ 905
4E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR14)	8	R/W	00h	23.5.15/ 905
4F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR15)	8	R/W	00h	23.5.15/ 905
50	INTC Software Set/Clear Interrupt Register (INTC_SSCIR16)	8	R/W	00h	23.5.15/ 905
51	INTC Software Set/Clear Interrupt Register (INTC_SSCIR17)	8	R/W	00h	23.5.15/ 905
52	INTC Software Set/Clear Interrupt Register (INTC_SSCIR18)	8	R/W	00h	23.5.15/ 905
53	INTC Software Set/Clear Interrupt Register (INTC_SSCIR19)	8	R/W	00h	23.5.15/ 905
54	INTC Software Set/Clear Interrupt Register (INTC_SSCIR20)	8	R/W	00h	23.5.15/ 905
55	INTC Software Set/Clear Interrupt Register (INTC_SSCIR21)	8	R/W	00h	23.5.15/ 905
56	INTC Software Set/Clear Interrupt Register (INTC_SSCIR22)	8	R/W	00h	23.5.15/ 905
57	INTC Software Set/Clear Interrupt Register (INTC_SSCIR23)	8	R/W	00h	23.5.15/ 905
58	INTC Software Set/Clear Interrupt Register (INTC_SSCIR24)	8	R/W	00h	23.5.15/ 905
59	INTC Software Set/Clear Interrupt Register (INTC_SSCIR25)	8	R/W	00h	23.5.15/ 905
5A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR26)	8	R/W	00h	23.5.15/ 905
5B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR27)	8	R/W	00h	23.5.15/ 905
5C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR28)	8	R/W	00h	23.5.15/ 905
5D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR29)	8	R/W	00h	23.5.15/ 905
5E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR30)	8	R/W	00h	23.5.15/ 905
5F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR31)	8	R/W	00h	23.5.15/ 905
60	INTC Priority Select Register (INTC_PSR0)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
62	INTC Priority Select Register (INTC_PSR1)	16	R/W	8000h	23.5.16/ 906
64	INTC Priority Select Register (INTC_PSR2)	16	R/W	8000h	23.5.16/ 906
66	INTC Priority Select Register (INTC_PSR3)	16	R/W	8000h	23.5.16/ 906
68	INTC Priority Select Register (INTC_PSR4)	16	R/W	8000h	23.5.16/ 906
6A	INTC Priority Select Register (INTC_PSR5)	16	R/W	8000h	23.5.16/ 906
6C	INTC Priority Select Register (INTC_PSR6)	16	R/W	8000h	23.5.16/ 906
6E	INTC Priority Select Register (INTC_PSR7)	16	R/W	8000h	23.5.16/ 906
70	INTC Priority Select Register (INTC_PSR8)	16	R/W	8000h	23.5.16/ 906
72	INTC Priority Select Register (INTC_PSR9)	16	R/W	8000h	23.5.16/ 906
74	INTC Priority Select Register (INTC_PSR10)	16	R/W	8000h	23.5.16/ 906
76	INTC Priority Select Register (INTC_PSR11)	16	R/W	8000h	23.5.16/ 906
78	INTC Priority Select Register (INTC_PSR12)	16	R/W	8000h	23.5.16/ 906
7A	INTC Priority Select Register (INTC_PSR13)	16	R/W	8000h	23.5.16/ 906
7C	INTC Priority Select Register (INTC_PSR14)	16	R/W	8000h	23.5.16/ 906
7E	INTC Priority Select Register (INTC_PSR15)	16	R/W	8000h	23.5.16/ 906
80	INTC Priority Select Register (INTC_PSR16)	16	R/W	8000h	23.5.16/ 906
82	INTC Priority Select Register (INTC_PSR17)	16	R/W	8000h	23.5.16/ 906
84	INTC Priority Select Register (INTC_PSR18)	16	R/W	8000h	23.5.16/ 906
86	INTC Priority Select Register (INTC_PSR19)	16	R/W	8000h	23.5.16/ 906
88	INTC Priority Select Register (INTC_PSR20)	16	R/W	8000h	23.5.16/ 906
8A	INTC Priority Select Register (INTC_PSR21)	16	R/W	8000h	23.5.16/ 906
8C	INTC Priority Select Register (INTC_PSR22)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8E	INTC Priority Select Register (INTC_PSR23)	16	R/W	8000h	23.5.16/ 906
90	INTC Priority Select Register (INTC_PSR24)	16	R/W	8000h	23.5.16/ 906
92	INTC Priority Select Register (INTC_PSR25)	16	R/W	8000h	23.5.16/ 906
94	INTC Priority Select Register (INTC_PSR26)	16	R/W	8000h	23.5.16/ 906
96	INTC Priority Select Register (INTC_PSR27)	16	R/W	8000h	23.5.16/ 906
98	INTC Priority Select Register (INTC_PSR28)	16	R/W	8000h	23.5.16/ 906
9A	INTC Priority Select Register (INTC_PSR29)	16	R/W	8000h	23.5.16/ 906
9C	INTC Priority Select Register (INTC_PSR30)	16	R/W	8000h	23.5.16/ 906
9E	INTC Priority Select Register (INTC_PSR31)	16	R/W	8000h	23.5.16/ 906
A0	INTC Priority Select Register (INTC_PSR32)	16	R/W	8000h	23.5.16/ 906
A2	INTC Priority Select Register (INTC_PSR33)	16	R/W	8000h	23.5.16/ 906
A4	INTC Priority Select Register (INTC_PSR34)	16	R/W	8000h	23.5.16/ 906
A6	INTC Priority Select Register (INTC_PSR35)	16	R/W	8000h	23.5.16/ 906
A8	INTC Priority Select Register (INTC_PSR36)	16	R/W	8000h	23.5.16/ 906
AA	INTC Priority Select Register (INTC_PSR37)	16	R/W	8000h	23.5.16/ 906
AC	INTC Priority Select Register (INTC_PSR38)	16	R/W	8000h	23.5.16/ 906
AE	INTC Priority Select Register (INTC_PSR39)	16	R/W	8000h	23.5.16/ 906
B0	INTC Priority Select Register (INTC_PSR40)	16	R/W	8000h	23.5.16/ 906
B2	INTC Priority Select Register (INTC_PSR41)	16	R/W	8000h	23.5.16/ 906
B4	INTC Priority Select Register (INTC_PSR42)	16	R/W	8000h	23.5.16/ 906
B6	INTC Priority Select Register (INTC_PSR43)	16	R/W	8000h	23.5.16/ 906
B8	INTC Priority Select Register (INTC_PSR44)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BA	INTC Priority Select Register (INTC_PSR45)	16	R/W	8000h	23.5.16/ 906
BC	INTC Priority Select Register (INTC_PSR46)	16	R/W	8000h	23.5.16/ 906
BE	INTC Priority Select Register (INTC_PSR47)	16	R/W	8000h	23.5.16/ 906
C0	INTC Priority Select Register (INTC_PSR48)	16	R/W	8000h	23.5.16/ 906
C2	INTC Priority Select Register (INTC_PSR49)	16	R/W	8000h	23.5.16/ 906
C4	INTC Priority Select Register (INTC_PSR50)	16	R/W	8000h	23.5.16/ 906
C6	INTC Priority Select Register (INTC_PSR51)	16	R/W	8000h	23.5.16/ 906
C8	INTC Priority Select Register (INTC_PSR52)	16	R/W	8000h	23.5.16/ 906
CA	INTC Priority Select Register (INTC_PSR53)	16	R/W	8000h	23.5.16/ 906
CC	INTC Priority Select Register (INTC_PSR54)	16	R/W	8000h	23.5.16/ 906
CE	INTC Priority Select Register (INTC_PSR55)	16	R/W	8000h	23.5.16/ 906
D0	INTC Priority Select Register (INTC_PSR56)	16	R/W	8000h	23.5.16/ 906
D2	INTC Priority Select Register (INTC_PSR57)	16	R/W	8000h	23.5.16/ 906
D4	INTC Priority Select Register (INTC_PSR58)	16	R/W	8000h	23.5.16/ 906
D6	INTC Priority Select Register (INTC_PSR59)	16	R/W	8000h	23.5.16/ 906
D8	INTC Priority Select Register (INTC_PSR60)	16	R/W	8000h	23.5.16/ 906
DA	INTC Priority Select Register (INTC_PSR61)	16	R/W	8000h	23.5.16/ 906
DC	INTC Priority Select Register (INTC_PSR62)	16	R/W	8000h	23.5.16/ 906
DE	INTC Priority Select Register (INTC_PSR63)	16	R/W	8000h	23.5.16/ 906
E0	INTC Priority Select Register (INTC_PSR64)	16	R/W	8000h	23.5.16/ 906
E2	INTC Priority Select Register (INTC_PSR65)	16	R/W	8000h	23.5.16/ 906
E4	INTC Priority Select Register (INTC_PSR66)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E6	INTC Priority Select Register (INTC_PSR67)	16	R/W	8000h	23.5.16/ 906
E8	INTC Priority Select Register (INTC_PSR68)	16	R/W	8000h	23.5.16/ 906
EA	INTC Priority Select Register (INTC_PSR69)	16	R/W	8000h	23.5.16/ 906
EC	INTC Priority Select Register (INTC_PSR70)	16	R/W	8000h	23.5.16/ 906
EE	INTC Priority Select Register (INTC_PSR71)	16	R/W	8000h	23.5.16/ 906
F0	INTC Priority Select Register (INTC_PSR72)	16	R/W	8000h	23.5.16/ 906
F2	INTC Priority Select Register (INTC_PSR73)	16	R/W	8000h	23.5.16/ 906
F4	INTC Priority Select Register (INTC_PSR74)	16	R/W	8000h	23.5.16/ 906
F6	INTC Priority Select Register (INTC_PSR75)	16	R/W	8000h	23.5.16/ 906
F8	INTC Priority Select Register (INTC_PSR76)	16	R/W	8000h	23.5.16/ 906
FA	INTC Priority Select Register (INTC_PSR77)	16	R/W	8000h	23.5.16/ 906
FC	INTC Priority Select Register (INTC_PSR78)	16	R/W	8000h	23.5.16/ 906
FE	INTC Priority Select Register (INTC_PSR79)	16	R/W	8000h	23.5.16/ 906
100	INTC Priority Select Register (INTC_PSR80)	16	R/W	8000h	23.5.16/ 906
102	INTC Priority Select Register (INTC_PSR81)	16	R/W	8000h	23.5.16/ 906
104	INTC Priority Select Register (INTC_PSR82)	16	R/W	8000h	23.5.16/ 906
106	INTC Priority Select Register (INTC_PSR83)	16	R/W	8000h	23.5.16/ 906
108	INTC Priority Select Register (INTC_PSR84)	16	R/W	8000h	23.5.16/ 906
10A	INTC Priority Select Register (INTC_PSR85)	16	R/W	8000h	23.5.16/ 906
10C	INTC Priority Select Register (INTC_PSR86)	16	R/W	8000h	23.5.16/ 906
10E	INTC Priority Select Register (INTC_PSR87)	16	R/W	8000h	23.5.16/ 906
110	INTC Priority Select Register (INTC_PSR88)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
112	INTC Priority Select Register (INTC_PSR89)	16	R/W	8000h	23.5.16/ 906
114	INTC Priority Select Register (INTC_PSR90)	16	R/W	8000h	23.5.16/ 906
116	INTC Priority Select Register (INTC_PSR91)	16	R/W	8000h	23.5.16/ 906
118	INTC Priority Select Register (INTC_PSR92)	16	R/W	8000h	23.5.16/ 906
11A	INTC Priority Select Register (INTC_PSR93)	16	R/W	8000h	23.5.16/ 906
11C	INTC Priority Select Register (INTC_PSR94)	16	R/W	8000h	23.5.16/ 906
11E	INTC Priority Select Register (INTC_PSR95)	16	R/W	8000h	23.5.16/ 906
120	INTC Priority Select Register (INTC_PSR96)	16	R/W	8000h	23.5.16/ 906
122	INTC Priority Select Register (INTC_PSR97)	16	R/W	8000h	23.5.16/ 906
124	INTC Priority Select Register (INTC_PSR98)	16	R/W	8000h	23.5.16/ 906
126	INTC Priority Select Register (INTC_PSR99)	16	R/W	8000h	23.5.16/ 906
128	INTC Priority Select Register (INTC_PSR100)	16	R/W	8000h	23.5.16/ 906
12A	INTC Priority Select Register (INTC_PSR101)	16	R/W	8000h	23.5.16/ 906
12C	INTC Priority Select Register (INTC_PSR102)	16	R/W	8000h	23.5.16/ 906
12E	INTC Priority Select Register (INTC_PSR103)	16	R/W	8000h	23.5.16/ 906
130	INTC Priority Select Register (INTC_PSR104)	16	R/W	8000h	23.5.16/ 906
132	INTC Priority Select Register (INTC_PSR105)	16	R/W	8000h	23.5.16/ 906
134	INTC Priority Select Register (INTC_PSR106)	16	R/W	8000h	23.5.16/ 906
136	INTC Priority Select Register (INTC_PSR107)	16	R/W	8000h	23.5.16/ 906
138	INTC Priority Select Register (INTC_PSR108)	16	R/W	8000h	23.5.16/ 906
13A	INTC Priority Select Register (INTC_PSR109)	16	R/W	8000h	23.5.16/ 906
13C	INTC Priority Select Register (INTC_PSR110)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13E	INTC Priority Select Register (INTC_PSR111)	16	R/W	8000h	23.5.16/ 906
140	INTC Priority Select Register (INTC_PSR112)	16	R/W	8000h	23.5.16/ 906
142	INTC Priority Select Register (INTC_PSR113)	16	R/W	8000h	23.5.16/ 906
144	INTC Priority Select Register (INTC_PSR114)	16	R/W	8000h	23.5.16/ 906
146	INTC Priority Select Register (INTC_PSR115)	16	R/W	8000h	23.5.16/ 906
148	INTC Priority Select Register (INTC_PSR116)	16	R/W	8000h	23.5.16/ 906
14A	INTC Priority Select Register (INTC_PSR117)	16	R/W	8000h	23.5.16/ 906
14C	INTC Priority Select Register (INTC_PSR118)	16	R/W	8000h	23.5.16/ 906
14E	INTC Priority Select Register (INTC_PSR119)	16	R/W	8000h	23.5.16/ 906
150	INTC Priority Select Register (INTC_PSR120)	16	R/W	8000h	23.5.16/ 906
152	INTC Priority Select Register (INTC_PSR121)	16	R/W	8000h	23.5.16/ 906
154	INTC Priority Select Register (INTC_PSR122)	16	R/W	8000h	23.5.16/ 906
156	INTC Priority Select Register (INTC_PSR123)	16	R/W	8000h	23.5.16/ 906
158	INTC Priority Select Register (INTC_PSR124)	16	R/W	8000h	23.5.16/ 906
15A	INTC Priority Select Register (INTC_PSR125)	16	R/W	8000h	23.5.16/ 906
15C	INTC Priority Select Register (INTC_PSR126)	16	R/W	8000h	23.5.16/ 906
15E	INTC Priority Select Register (INTC_PSR127)	16	R/W	8000h	23.5.16/ 906
160	INTC Priority Select Register (INTC_PSR128)	16	R/W	8000h	23.5.16/ 906
162	INTC Priority Select Register (INTC_PSR129)	16	R/W	8000h	23.5.16/ 906
164	INTC Priority Select Register (INTC_PSR130)	16	R/W	8000h	23.5.16/ 906
166	INTC Priority Select Register (INTC_PSR131)	16	R/W	8000h	23.5.16/ 906
168	INTC Priority Select Register (INTC_PSR132)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16A	INTC Priority Select Register (INTC_PSR133)	16	R/W	8000h	23.5.16/ 906
16C	INTC Priority Select Register (INTC_PSR134)	16	R/W	8000h	23.5.16/ 906
16E	INTC Priority Select Register (INTC_PSR135)	16	R/W	8000h	23.5.16/ 906
170	INTC Priority Select Register (INTC_PSR136)	16	R/W	8000h	23.5.16/ 906
172	INTC Priority Select Register (INTC_PSR137)	16	R/W	8000h	23.5.16/ 906
174	INTC Priority Select Register (INTC_PSR138)	16	R/W	8000h	23.5.16/ 906
176	INTC Priority Select Register (INTC_PSR139)	16	R/W	8000h	23.5.16/ 906
178	INTC Priority Select Register (INTC_PSR140)	16	R/W	8000h	23.5.16/ 906
17A	INTC Priority Select Register (INTC_PSR141)	16	R/W	8000h	23.5.16/ 906
17C	INTC Priority Select Register (INTC_PSR142)	16	R/W	8000h	23.5.16/ 906
17E	INTC Priority Select Register (INTC_PSR143)	16	R/W	8000h	23.5.16/ 906
180	INTC Priority Select Register (INTC_PSR144)	16	R/W	8000h	23.5.16/ 906
182	INTC Priority Select Register (INTC_PSR145)	16	R/W	8000h	23.5.16/ 906
184	INTC Priority Select Register (INTC_PSR146)	16	R/W	8000h	23.5.16/ 906
186	INTC Priority Select Register (INTC_PSR147)	16	R/W	8000h	23.5.16/ 906
188	INTC Priority Select Register (INTC_PSR148)	16	R/W	8000h	23.5.16/ 906
18A	INTC Priority Select Register (INTC_PSR149)	16	R/W	8000h	23.5.16/ 906
18C	INTC Priority Select Register (INTC_PSR150)	16	R/W	8000h	23.5.16/ 906
18E	INTC Priority Select Register (INTC_PSR151)	16	R/W	8000h	23.5.16/ 906
190	INTC Priority Select Register (INTC_PSR152)	16	R/W	8000h	23.5.16/ 906
192	INTC Priority Select Register (INTC_PSR153)	16	R/W	8000h	23.5.16/ 906
194	INTC Priority Select Register (INTC_PSR154)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
196	INTC Priority Select Register (INTC_PSR155)	16	R/W	8000h	23.5.16/ 906
198	INTC Priority Select Register (INTC_PSR156)	16	R/W	8000h	23.5.16/ 906
19A	INTC Priority Select Register (INTC_PSR157)	16	R/W	8000h	23.5.16/ 906
19C	INTC Priority Select Register (INTC_PSR158)	16	R/W	8000h	23.5.16/ 906
19E	INTC Priority Select Register (INTC_PSR159)	16	R/W	8000h	23.5.16/ 906
1A0	INTC Priority Select Register (INTC_PSR160)	16	R/W	8000h	23.5.16/ 906
1A2	INTC Priority Select Register (INTC_PSR161)	16	R/W	8000h	23.5.16/ 906
1A4	INTC Priority Select Register (INTC_PSR162)	16	R/W	8000h	23.5.16/ 906
1A6	INTC Priority Select Register (INTC_PSR163)	16	R/W	8000h	23.5.16/ 906
1A8	INTC Priority Select Register (INTC_PSR164)	16	R/W	8000h	23.5.16/ 906
1AA	INTC Priority Select Register (INTC_PSR165)	16	R/W	8000h	23.5.16/ 906
1AC	INTC Priority Select Register (INTC_PSR166)	16	R/W	8000h	23.5.16/ 906
1AE	INTC Priority Select Register (INTC_PSR167)	16	R/W	8000h	23.5.16/ 906
1B0	INTC Priority Select Register (INTC_PSR168)	16	R/W	8000h	23.5.16/ 906
1B2	INTC Priority Select Register (INTC_PSR169)	16	R/W	8000h	23.5.16/ 906
1B4	INTC Priority Select Register (INTC_PSR170)	16	R/W	8000h	23.5.16/ 906
1B6	INTC Priority Select Register (INTC_PSR171)	16	R/W	8000h	23.5.16/ 906
1B8	INTC Priority Select Register (INTC_PSR172)	16	R/W	8000h	23.5.16/ 906
1BA	INTC Priority Select Register (INTC_PSR173)	16	R/W	8000h	23.5.16/ 906
1BC	INTC Priority Select Register (INTC_PSR174)	16	R/W	8000h	23.5.16/ 906
1BE	INTC Priority Select Register (INTC_PSR175)	16	R/W	8000h	23.5.16/ 906
1C0	INTC Priority Select Register (INTC_PSR176)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C2	INTC Priority Select Register (INTC_PSR177)	16	R/W	8000h	23.5.16/ 906
1C4	INTC Priority Select Register (INTC_PSR178)	16	R/W	8000h	23.5.16/ 906
1C6	INTC Priority Select Register (INTC_PSR179)	16	R/W	8000h	23.5.16/ 906
1C8	INTC Priority Select Register (INTC_PSR180)	16	R/W	8000h	23.5.16/ 906
1CA	INTC Priority Select Register (INTC_PSR181)	16	R/W	8000h	23.5.16/ 906
1CC	INTC Priority Select Register (INTC_PSR182)	16	R/W	8000h	23.5.16/ 906
1CE	INTC Priority Select Register (INTC_PSR183)	16	R/W	8000h	23.5.16/ 906
1D0	INTC Priority Select Register (INTC_PSR184)	16	R/W	8000h	23.5.16/ 906
1D2	INTC Priority Select Register (INTC_PSR185)	16	R/W	8000h	23.5.16/ 906
1D4	INTC Priority Select Register (INTC_PSR186)	16	R/W	8000h	23.5.16/ 906
1D6	INTC Priority Select Register (INTC_PSR187)	16	R/W	8000h	23.5.16/ 906
1D8	INTC Priority Select Register (INTC_PSR188)	16	R/W	8000h	23.5.16/ 906
1DA	INTC Priority Select Register (INTC_PSR189)	16	R/W	8000h	23.5.16/ 906
1DC	INTC Priority Select Register (INTC_PSR190)	16	R/W	8000h	23.5.16/ 906
1DE	INTC Priority Select Register (INTC_PSR191)	16	R/W	8000h	23.5.16/ 906
1E0	INTC Priority Select Register (INTC_PSR192)	16	R/W	8000h	23.5.16/ 906
1E2	INTC Priority Select Register (INTC_PSR193)	16	R/W	8000h	23.5.16/ 906
1E4	INTC Priority Select Register (INTC_PSR194)	16	R/W	8000h	23.5.16/ 906
1E6	INTC Priority Select Register (INTC_PSR195)	16	R/W	8000h	23.5.16/ 906
1E8	INTC Priority Select Register (INTC_PSR196)	16	R/W	8000h	23.5.16/ 906
1EA	INTC Priority Select Register (INTC_PSR197)	16	R/W	8000h	23.5.16/ 906
1EC	INTC Priority Select Register (INTC_PSR198)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1EE	INTC Priority Select Register (INTC_PSR199)	16	R/W	8000h	23.5.16/ 906
1F0	INTC Priority Select Register (INTC_PSR200)	16	R/W	8000h	23.5.16/ 906
1F2	INTC Priority Select Register (INTC_PSR201)	16	R/W	8000h	23.5.16/ 906
1F4	INTC Priority Select Register (INTC_PSR202)	16	R/W	8000h	23.5.16/ 906
1F6	INTC Priority Select Register (INTC_PSR203)	16	R/W	8000h	23.5.16/ 906
1F8	INTC Priority Select Register (INTC_PSR204)	16	R/W	8000h	23.5.16/ 906
1FA	INTC Priority Select Register (INTC_PSR205)	16	R/W	8000h	23.5.16/ 906
1FC	INTC Priority Select Register (INTC_PSR206)	16	R/W	8000h	23.5.16/ 906
1FE	INTC Priority Select Register (INTC_PSR207)	16	R/W	8000h	23.5.16/ 906
200	INTC Priority Select Register (INTC_PSR208)	16	R/W	8000h	23.5.16/ 906
202	INTC Priority Select Register (INTC_PSR209)	16	R/W	8000h	23.5.16/ 906
204	INTC Priority Select Register (INTC_PSR210)	16	R/W	8000h	23.5.16/ 906
206	INTC Priority Select Register (INTC_PSR211)	16	R/W	8000h	23.5.16/ 906
208	INTC Priority Select Register (INTC_PSR212)	16	R/W	8000h	23.5.16/ 906
20A	INTC Priority Select Register (INTC_PSR213)	16	R/W	8000h	23.5.16/ 906
20C	INTC Priority Select Register (INTC_PSR214)	16	R/W	8000h	23.5.16/ 906
20E	INTC Priority Select Register (INTC_PSR215)	16	R/W	8000h	23.5.16/ 906
210	INTC Priority Select Register (INTC_PSR216)	16	R/W	8000h	23.5.16/ 906
212	INTC Priority Select Register (INTC_PSR217)	16	R/W	8000h	23.5.16/ 906
214	INTC Priority Select Register (INTC_PSR218)	16	R/W	8000h	23.5.16/ 906
216	INTC Priority Select Register (INTC_PSR219)	16	R/W	8000h	23.5.16/ 906
218	INTC Priority Select Register (INTC_PSR220)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21A	INTC Priority Select Register (INTC_PSR221)	16	R/W	8000h	23.5.16/ 906
21C	INTC Priority Select Register (INTC_PSR222)	16	R/W	8000h	23.5.16/ 906
21E	INTC Priority Select Register (INTC_PSR223)	16	R/W	8000h	23.5.16/ 906
220	INTC Priority Select Register (INTC_PSR224)	16	R/W	8000h	23.5.16/ 906
222	INTC Priority Select Register (INTC_PSR225)	16	R/W	8000h	23.5.16/ 906
224	INTC Priority Select Register (INTC_PSR226)	16	R/W	8000h	23.5.16/ 906
226	INTC Priority Select Register (INTC_PSR227)	16	R/W	8000h	23.5.16/ 906
228	INTC Priority Select Register (INTC_PSR228)	16	R/W	8000h	23.5.16/ 906
22A	INTC Priority Select Register (INTC_PSR229)	16	R/W	8000h	23.5.16/ 906
22C	INTC Priority Select Register (INTC_PSR230)	16	R/W	8000h	23.5.16/ 906
22E	INTC Priority Select Register (INTC_PSR231)	16	R/W	8000h	23.5.16/ 906
230	INTC Priority Select Register (INTC_PSR232)	16	R/W	8000h	23.5.16/ 906
232	INTC Priority Select Register (INTC_PSR233)	16	R/W	8000h	23.5.16/ 906
234	INTC Priority Select Register (INTC_PSR234)	16	R/W	8000h	23.5.16/ 906
236	INTC Priority Select Register (INTC_PSR235)	16	R/W	8000h	23.5.16/ 906
238	INTC Priority Select Register (INTC_PSR236)	16	R/W	8000h	23.5.16/ 906
23A	INTC Priority Select Register (INTC_PSR237)	16	R/W	8000h	23.5.16/ 906
23C	INTC Priority Select Register (INTC_PSR238)	16	R/W	8000h	23.5.16/ 906
23E	INTC Priority Select Register (INTC_PSR239)	16	R/W	8000h	23.5.16/ 906
240	INTC Priority Select Register (INTC_PSR240)	16	R/W	8000h	23.5.16/ 906
242	INTC Priority Select Register (INTC_PSR241)	16	R/W	8000h	23.5.16/ 906
244	INTC Priority Select Register (INTC_PSR242)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
246	INTC Priority Select Register (INTC_PSR243)	16	R/W	8000h	23.5.16/ 906
248	INTC Priority Select Register (INTC_PSR244)	16	R/W	8000h	23.5.16/ 906
24A	INTC Priority Select Register (INTC_PSR245)	16	R/W	8000h	23.5.16/ 906
24C	INTC Priority Select Register (INTC_PSR246)	16	R/W	8000h	23.5.16/ 906
24E	INTC Priority Select Register (INTC_PSR247)	16	R/W	8000h	23.5.16/ 906
250	INTC Priority Select Register (INTC_PSR248)	16	R/W	8000h	23.5.16/ 906
252	INTC Priority Select Register (INTC_PSR249)	16	R/W	8000h	23.5.16/ 906
254	INTC Priority Select Register (INTC_PSR250)	16	R/W	8000h	23.5.16/ 906
256	INTC Priority Select Register (INTC_PSR251)	16	R/W	8000h	23.5.16/ 906
258	INTC Priority Select Register (INTC_PSR252)	16	R/W	8000h	23.5.16/ 906
25A	INTC Priority Select Register (INTC_PSR253)	16	R/W	8000h	23.5.16/ 906
25C	INTC Priority Select Register (INTC_PSR254)	16	R/W	8000h	23.5.16/ 906
25E	INTC Priority Select Register (INTC_PSR255)	16	R/W	8000h	23.5.16/ 906
260	INTC Priority Select Register (INTC_PSR256)	16	R/W	8000h	23.5.16/ 906
262	INTC Priority Select Register (INTC_PSR257)	16	R/W	8000h	23.5.16/ 906
264	INTC Priority Select Register (INTC_PSR258)	16	R/W	8000h	23.5.16/ 906
266	INTC Priority Select Register (INTC_PSR259)	16	R/W	8000h	23.5.16/ 906
268	INTC Priority Select Register (INTC_PSR260)	16	R/W	8000h	23.5.16/ 906
26A	INTC Priority Select Register (INTC_PSR261)	16	R/W	8000h	23.5.16/ 906
26C	INTC Priority Select Register (INTC_PSR262)	16	R/W	8000h	23.5.16/ 906
26E	INTC Priority Select Register (INTC_PSR263)	16	R/W	8000h	23.5.16/ 906
270	INTC Priority Select Register (INTC_PSR264)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
272	INTC Priority Select Register (INTC_PSR265)	16	R/W	8000h	23.5.16/ 906
274	INTC Priority Select Register (INTC_PSR266)	16	R/W	8000h	23.5.16/ 906
276	INTC Priority Select Register (INTC_PSR267)	16	R/W	8000h	23.5.16/ 906
278	INTC Priority Select Register (INTC_PSR268)	16	R/W	8000h	23.5.16/ 906
27A	INTC Priority Select Register (INTC_PSR269)	16	R/W	8000h	23.5.16/ 906
27C	INTC Priority Select Register (INTC_PSR270)	16	R/W	8000h	23.5.16/ 906
27E	INTC Priority Select Register (INTC_PSR271)	16	R/W	8000h	23.5.16/ 906
280	INTC Priority Select Register (INTC_PSR272)	16	R/W	8000h	23.5.16/ 906
282	INTC Priority Select Register (INTC_PSR273)	16	R/W	8000h	23.5.16/ 906
284	INTC Priority Select Register (INTC_PSR274)	16	R/W	8000h	23.5.16/ 906
286	INTC Priority Select Register (INTC_PSR275)	16	R/W	8000h	23.5.16/ 906
288	INTC Priority Select Register (INTC_PSR276)	16	R/W	8000h	23.5.16/ 906
28A	INTC Priority Select Register (INTC_PSR277)	16	R/W	8000h	23.5.16/ 906
28C	INTC Priority Select Register (INTC_PSR278)	16	R/W	8000h	23.5.16/ 906
28E	INTC Priority Select Register (INTC_PSR279)	16	R/W	8000h	23.5.16/ 906
290	INTC Priority Select Register (INTC_PSR280)	16	R/W	8000h	23.5.16/ 906
292	INTC Priority Select Register (INTC_PSR281)	16	R/W	8000h	23.5.16/ 906
294	INTC Priority Select Register (INTC_PSR282)	16	R/W	8000h	23.5.16/ 906
296	INTC Priority Select Register (INTC_PSR283)	16	R/W	8000h	23.5.16/ 906
298	INTC Priority Select Register (INTC_PSR284)	16	R/W	8000h	23.5.16/ 906
29A	INTC Priority Select Register (INTC_PSR285)	16	R/W	8000h	23.5.16/ 906
29C	INTC Priority Select Register (INTC_PSR286)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29E	INTC Priority Select Register (INTC_PSR287)	16	R/W	8000h	23.5.16/ 906
2A0	INTC Priority Select Register (INTC_PSR288)	16	R/W	8000h	23.5.16/ 906
2A2	INTC Priority Select Register (INTC_PSR289)	16	R/W	8000h	23.5.16/ 906
2A4	INTC Priority Select Register (INTC_PSR290)	16	R/W	8000h	23.5.16/ 906
2A6	INTC Priority Select Register (INTC_PSR291)	16	R/W	8000h	23.5.16/ 906
2A8	INTC Priority Select Register (INTC_PSR292)	16	R/W	8000h	23.5.16/ 906
2AA	INTC Priority Select Register (INTC_PSR293)	16	R/W	8000h	23.5.16/ 906
2AC	INTC Priority Select Register (INTC_PSR294)	16	R/W	8000h	23.5.16/ 906
2AE	INTC Priority Select Register (INTC_PSR295)	16	R/W	8000h	23.5.16/ 906
2B0	INTC Priority Select Register (INTC_PSR296)	16	R/W	8000h	23.5.16/ 906
2B2	INTC Priority Select Register (INTC_PSR297)	16	R/W	8000h	23.5.16/ 906
2B4	INTC Priority Select Register (INTC_PSR298)	16	R/W	8000h	23.5.16/ 906
2B6	INTC Priority Select Register (INTC_PSR299)	16	R/W	8000h	23.5.16/ 906
2B8	INTC Priority Select Register (INTC_PSR300)	16	R/W	8000h	23.5.16/ 906
2BA	INTC Priority Select Register (INTC_PSR301)	16	R/W	8000h	23.5.16/ 906
2BC	INTC Priority Select Register (INTC_PSR302)	16	R/W	8000h	23.5.16/ 906
2BE	INTC Priority Select Register (INTC_PSR303)	16	R/W	8000h	23.5.16/ 906
2C0	INTC Priority Select Register (INTC_PSR304)	16	R/W	8000h	23.5.16/ 906
2C2	INTC Priority Select Register (INTC_PSR305)	16	R/W	8000h	23.5.16/ 906
2C4	INTC Priority Select Register (INTC_PSR306)	16	R/W	8000h	23.5.16/ 906
2C6	INTC Priority Select Register (INTC_PSR307)	16	R/W	8000h	23.5.16/ 906
2C8	INTC Priority Select Register (INTC_PSR308)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2CA	INTC Priority Select Register (INTC_PSR309)	16	R/W	8000h	23.5.16/ 906
2CC	INTC Priority Select Register (INTC_PSR310)	16	R/W	8000h	23.5.16/ 906
2CE	INTC Priority Select Register (INTC_PSR311)	16	R/W	8000h	23.5.16/ 906
2D0	INTC Priority Select Register (INTC_PSR312)	16	R/W	8000h	23.5.16/ 906
2D2	INTC Priority Select Register (INTC_PSR313)	16	R/W	8000h	23.5.16/ 906
2D4	INTC Priority Select Register (INTC_PSR314)	16	R/W	8000h	23.5.16/ 906
2D6	INTC Priority Select Register (INTC_PSR315)	16	R/W	8000h	23.5.16/ 906
2D8	INTC Priority Select Register (INTC_PSR316)	16	R/W	8000h	23.5.16/ 906
2DA	INTC Priority Select Register (INTC_PSR317)	16	R/W	8000h	23.5.16/ 906
2DC	INTC Priority Select Register (INTC_PSR318)	16	R/W	8000h	23.5.16/ 906
2DE	INTC Priority Select Register (INTC_PSR319)	16	R/W	8000h	23.5.16/ 906
2E0	INTC Priority Select Register (INTC_PSR320)	16	R/W	8000h	23.5.16/ 906
2E2	INTC Priority Select Register (INTC_PSR321)	16	R/W	8000h	23.5.16/ 906
2E4	INTC Priority Select Register (INTC_PSR322)	16	R/W	8000h	23.5.16/ 906
2E6	INTC Priority Select Register (INTC_PSR323)	16	R/W	8000h	23.5.16/ 906
2E8	INTC Priority Select Register (INTC_PSR324)	16	R/W	8000h	23.5.16/ 906
2EA	INTC Priority Select Register (INTC_PSR325)	16	R/W	8000h	23.5.16/ 906
2EC	INTC Priority Select Register (INTC_PSR326)	16	R/W	8000h	23.5.16/ 906
2EE	INTC Priority Select Register (INTC_PSR327)	16	R/W	8000h	23.5.16/ 906
2F0	INTC Priority Select Register (INTC_PSR328)	16	R/W	8000h	23.5.16/ 906
2F2	INTC Priority Select Register (INTC_PSR329)	16	R/W	8000h	23.5.16/ 906
2F4	INTC Priority Select Register (INTC_PSR330)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2F6	INTC Priority Select Register (INTC_PSR331)	16	R/W	8000h	23.5.16/ 906
2F8	INTC Priority Select Register (INTC_PSR332)	16	R/W	8000h	23.5.16/ 906
2FA	INTC Priority Select Register (INTC_PSR333)	16	R/W	8000h	23.5.16/ 906
2FC	INTC Priority Select Register (INTC_PSR334)	16	R/W	8000h	23.5.16/ 906
2FE	INTC Priority Select Register (INTC_PSR335)	16	R/W	8000h	23.5.16/ 906
300	INTC Priority Select Register (INTC_PSR336)	16	R/W	8000h	23.5.16/ 906
302	INTC Priority Select Register (INTC_PSR337)	16	R/W	8000h	23.5.16/ 906
304	INTC Priority Select Register (INTC_PSR338)	16	R/W	8000h	23.5.16/ 906
306	INTC Priority Select Register (INTC_PSR339)	16	R/W	8000h	23.5.16/ 906
308	INTC Priority Select Register (INTC_PSR340)	16	R/W	8000h	23.5.16/ 906
30A	INTC Priority Select Register (INTC_PSR341)	16	R/W	8000h	23.5.16/ 906
30C	INTC Priority Select Register (INTC_PSR342)	16	R/W	8000h	23.5.16/ 906
30E	INTC Priority Select Register (INTC_PSR343)	16	R/W	8000h	23.5.16/ 906
310	INTC Priority Select Register (INTC_PSR344)	16	R/W	8000h	23.5.16/ 906
312	INTC Priority Select Register (INTC_PSR345)	16	R/W	8000h	23.5.16/ 906
314	INTC Priority Select Register (INTC_PSR346)	16	R/W	8000h	23.5.16/ 906
316	INTC Priority Select Register (INTC_PSR347)	16	R/W	8000h	23.5.16/ 906
318	INTC Priority Select Register (INTC_PSR348)	16	R/W	8000h	23.5.16/ 906
31A	INTC Priority Select Register (INTC_PSR349)	16	R/W	8000h	23.5.16/ 906
31C	INTC Priority Select Register (INTC_PSR350)	16	R/W	8000h	23.5.16/ 906
31E	INTC Priority Select Register (INTC_PSR351)	16	R/W	8000h	23.5.16/ 906
320	INTC Priority Select Register (INTC_PSR352)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
322	INTC Priority Select Register (INTC_PSR353)	16	R/W	8000h	23.5.16/ 906
324	INTC Priority Select Register (INTC_PSR354)	16	R/W	8000h	23.5.16/ 906
326	INTC Priority Select Register (INTC_PSR355)	16	R/W	8000h	23.5.16/ 906
328	INTC Priority Select Register (INTC_PSR356)	16	R/W	8000h	23.5.16/ 906
32A	INTC Priority Select Register (INTC_PSR357)	16	R/W	8000h	23.5.16/ 906
32C	INTC Priority Select Register (INTC_PSR358)	16	R/W	8000h	23.5.16/ 906
32E	INTC Priority Select Register (INTC_PSR359)	16	R/W	8000h	23.5.16/ 906
330	INTC Priority Select Register (INTC_PSR360)	16	R/W	8000h	23.5.16/ 906
332	INTC Priority Select Register (INTC_PSR361)	16	R/W	8000h	23.5.16/ 906
334	INTC Priority Select Register (INTC_PSR362)	16	R/W	8000h	23.5.16/ 906
336	INTC Priority Select Register (INTC_PSR363)	16	R/W	8000h	23.5.16/ 906
338	INTC Priority Select Register (INTC_PSR364)	16	R/W	8000h	23.5.16/ 906
33A	INTC Priority Select Register (INTC_PSR365)	16	R/W	8000h	23.5.16/ 906
33C	INTC Priority Select Register (INTC_PSR366)	16	R/W	8000h	23.5.16/ 906
33E	INTC Priority Select Register (INTC_PSR367)	16	R/W	8000h	23.5.16/ 906
340	INTC Priority Select Register (INTC_PSR368)	16	R/W	8000h	23.5.16/ 906
342	INTC Priority Select Register (INTC_PSR369)	16	R/W	8000h	23.5.16/ 906
344	INTC Priority Select Register (INTC_PSR370)	16	R/W	8000h	23.5.16/ 906
346	INTC Priority Select Register (INTC_PSR371)	16	R/W	8000h	23.5.16/ 906
348	INTC Priority Select Register (INTC_PSR372)	16	R/W	8000h	23.5.16/ 906
34A	INTC Priority Select Register (INTC_PSR373)	16	R/W	8000h	23.5.16/ 906
34C	INTC Priority Select Register (INTC_PSR374)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34E	INTC Priority Select Register (INTC_PSR375)	16	R/W	8000h	23.5.16/ 906
350	INTC Priority Select Register (INTC_PSR376)	16	R/W	8000h	23.5.16/ 906
352	INTC Priority Select Register (INTC_PSR377)	16	R/W	8000h	23.5.16/ 906
354	INTC Priority Select Register (INTC_PSR378)	16	R/W	8000h	23.5.16/ 906
356	INTC Priority Select Register (INTC_PSR379)	16	R/W	8000h	23.5.16/ 906
358	INTC Priority Select Register (INTC_PSR380)	16	R/W	8000h	23.5.16/ 906
35A	INTC Priority Select Register (INTC_PSR381)	16	R/W	8000h	23.5.16/ 906
35C	INTC Priority Select Register (INTC_PSR382)	16	R/W	8000h	23.5.16/ 906
35E	INTC Priority Select Register (INTC_PSR383)	16	R/W	8000h	23.5.16/ 906
360	INTC Priority Select Register (INTC_PSR384)	16	R/W	8000h	23.5.16/ 906
362	INTC Priority Select Register (INTC_PSR385)	16	R/W	8000h	23.5.16/ 906
364	INTC Priority Select Register (INTC_PSR386)	16	R/W	8000h	23.5.16/ 906
366	INTC Priority Select Register (INTC_PSR387)	16	R/W	8000h	23.5.16/ 906
368	INTC Priority Select Register (INTC_PSR388)	16	R/W	8000h	23.5.16/ 906
36A	INTC Priority Select Register (INTC_PSR389)	16	R/W	8000h	23.5.16/ 906
36C	INTC Priority Select Register (INTC_PSR390)	16	R/W	8000h	23.5.16/ 906
36E	INTC Priority Select Register (INTC_PSR391)	16	R/W	8000h	23.5.16/ 906
370	INTC Priority Select Register (INTC_PSR392)	16	R/W	8000h	23.5.16/ 906
372	INTC Priority Select Register (INTC_PSR393)	16	R/W	8000h	23.5.16/ 906
374	INTC Priority Select Register (INTC_PSR394)	16	R/W	8000h	23.5.16/ 906
376	INTC Priority Select Register (INTC_PSR395)	16	R/W	8000h	23.5.16/ 906
378	INTC Priority Select Register (INTC_PSR396)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
37A	INTC Priority Select Register (INTC_PSR397)	16	R/W	8000h	23.5.16/ 906
37C	INTC Priority Select Register (INTC_PSR398)	16	R/W	8000h	23.5.16/ 906
37E	INTC Priority Select Register (INTC_PSR399)	16	R/W	8000h	23.5.16/ 906
380	INTC Priority Select Register (INTC_PSR400)	16	R/W	8000h	23.5.16/ 906
382	INTC Priority Select Register (INTC_PSR401)	16	R/W	8000h	23.5.16/ 906
384	INTC Priority Select Register (INTC_PSR402)	16	R/W	8000h	23.5.16/ 906
386	INTC Priority Select Register (INTC_PSR403)	16	R/W	8000h	23.5.16/ 906
388	INTC Priority Select Register (INTC_PSR404)	16	R/W	8000h	23.5.16/ 906
38A	INTC Priority Select Register (INTC_PSR405)	16	R/W	8000h	23.5.16/ 906
38C	INTC Priority Select Register (INTC_PSR406)	16	R/W	8000h	23.5.16/ 906
38E	INTC Priority Select Register (INTC_PSR407)	16	R/W	8000h	23.5.16/ 906
390	INTC Priority Select Register (INTC_PSR408)	16	R/W	8000h	23.5.16/ 906
392	INTC Priority Select Register (INTC_PSR409)	16	R/W	8000h	23.5.16/ 906
394	INTC Priority Select Register (INTC_PSR410)	16	R/W	8000h	23.5.16/ 906
396	INTC Priority Select Register (INTC_PSR411)	16	R/W	8000h	23.5.16/ 906
398	INTC Priority Select Register (INTC_PSR412)	16	R/W	8000h	23.5.16/ 906
39A	INTC Priority Select Register (INTC_PSR413)	16	R/W	8000h	23.5.16/ 906
39C	INTC Priority Select Register (INTC_PSR414)	16	R/W	8000h	23.5.16/ 906
39E	INTC Priority Select Register (INTC_PSR415)	16	R/W	8000h	23.5.16/ 906
3A0	INTC Priority Select Register (INTC_PSR416)	16	R/W	8000h	23.5.16/ 906
3A2	INTC Priority Select Register (INTC_PSR417)	16	R/W	8000h	23.5.16/ 906
3A4	INTC Priority Select Register (INTC_PSR418)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3A6	INTC Priority Select Register (INTC_PSR419)	16	R/W	8000h	23.5.16/ 906
3A8	INTC Priority Select Register (INTC_PSR420)	16	R/W	8000h	23.5.16/ 906
3AA	INTC Priority Select Register (INTC_PSR421)	16	R/W	8000h	23.5.16/ 906
3AC	INTC Priority Select Register (INTC_PSR422)	16	R/W	8000h	23.5.16/ 906
3AE	INTC Priority Select Register (INTC_PSR423)	16	R/W	8000h	23.5.16/ 906
3B0	INTC Priority Select Register (INTC_PSR424)	16	R/W	8000h	23.5.16/ 906
3B2	INTC Priority Select Register (INTC_PSR425)	16	R/W	8000h	23.5.16/ 906
3B4	INTC Priority Select Register (INTC_PSR426)	16	R/W	8000h	23.5.16/ 906
3B6	INTC Priority Select Register (INTC_PSR427)	16	R/W	8000h	23.5.16/ 906
3B8	INTC Priority Select Register (INTC_PSR428)	16	R/W	8000h	23.5.16/ 906
3BA	INTC Priority Select Register (INTC_PSR429)	16	R/W	8000h	23.5.16/ 906
3BC	INTC Priority Select Register (INTC_PSR430)	16	R/W	8000h	23.5.16/ 906
3BE	INTC Priority Select Register (INTC_PSR431)	16	R/W	8000h	23.5.16/ 906
3C0	INTC Priority Select Register (INTC_PSR432)	16	R/W	8000h	23.5.16/ 906
3C2	INTC Priority Select Register (INTC_PSR433)	16	R/W	8000h	23.5.16/ 906
3C4	INTC Priority Select Register (INTC_PSR434)	16	R/W	8000h	23.5.16/ 906
3C6	INTC Priority Select Register (INTC_PSR435)	16	R/W	8000h	23.5.16/ 906
3C8	INTC Priority Select Register (INTC_PSR436)	16	R/W	8000h	23.5.16/ 906
3CA	INTC Priority Select Register (INTC_PSR437)	16	R/W	8000h	23.5.16/ 906
3CC	INTC Priority Select Register (INTC_PSR438)	16	R/W	8000h	23.5.16/ 906
3CE	INTC Priority Select Register (INTC_PSR439)	16	R/W	8000h	23.5.16/ 906
3D0	INTC Priority Select Register (INTC_PSR440)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3D2	INTC Priority Select Register (INTC_PSR441)	16	R/W	8000h	23.5.16/ 906
3D4	INTC Priority Select Register (INTC_PSR442)	16	R/W	8000h	23.5.16/ 906
3D6	INTC Priority Select Register (INTC_PSR443)	16	R/W	8000h	23.5.16/ 906
3D8	INTC Priority Select Register (INTC_PSR444)	16	R/W	8000h	23.5.16/ 906
3DA	INTC Priority Select Register (INTC_PSR445)	16	R/W	8000h	23.5.16/ 906
3DC	INTC Priority Select Register (INTC_PSR446)	16	R/W	8000h	23.5.16/ 906
3DE	INTC Priority Select Register (INTC_PSR447)	16	R/W	8000h	23.5.16/ 906
3E0	INTC Priority Select Register (INTC_PSR448)	16	R/W	8000h	23.5.16/ 906
3E2	INTC Priority Select Register (INTC_PSR449)	16	R/W	8000h	23.5.16/ 906
3E4	INTC Priority Select Register (INTC_PSR450)	16	R/W	8000h	23.5.16/ 906
3E6	INTC Priority Select Register (INTC_PSR451)	16	R/W	8000h	23.5.16/ 906
3E8	INTC Priority Select Register (INTC_PSR452)	16	R/W	8000h	23.5.16/ 906
3EA	INTC Priority Select Register (INTC_PSR453)	16	R/W	8000h	23.5.16/ 906
3EC	INTC Priority Select Register (INTC_PSR454)	16	R/W	8000h	23.5.16/ 906
3EE	INTC Priority Select Register (INTC_PSR455)	16	R/W	8000h	23.5.16/ 906
3F0	INTC Priority Select Register (INTC_PSR456)	16	R/W	8000h	23.5.16/ 906
3F2	INTC Priority Select Register (INTC_PSR457)	16	R/W	8000h	23.5.16/ 906
3F4	INTC Priority Select Register (INTC_PSR458)	16	R/W	8000h	23.5.16/ 906
3F6	INTC Priority Select Register (INTC_PSR459)	16	R/W	8000h	23.5.16/ 906
3F8	INTC Priority Select Register (INTC_PSR460)	16	R/W	8000h	23.5.16/ 906
3FA	INTC Priority Select Register (INTC_PSR461)	16	R/W	8000h	23.5.16/ 906
3FC	INTC Priority Select Register (INTC_PSR462)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3FE	INTC Priority Select Register (INTC_PSR463)	16	R/W	8000h	23.5.16/ 906
400	INTC Priority Select Register (INTC_PSR464)	16	R/W	8000h	23.5.16/ 906
402	INTC Priority Select Register (INTC_PSR465)	16	R/W	8000h	23.5.16/ 906
404	INTC Priority Select Register (INTC_PSR466)	16	R/W	8000h	23.5.16/ 906
406	INTC Priority Select Register (INTC_PSR467)	16	R/W	8000h	23.5.16/ 906
408	INTC Priority Select Register (INTC_PSR468)	16	R/W	8000h	23.5.16/ 906
40A	INTC Priority Select Register (INTC_PSR469)	16	R/W	8000h	23.5.16/ 906
40C	INTC Priority Select Register (INTC_PSR470)	16	R/W	8000h	23.5.16/ 906
40E	INTC Priority Select Register (INTC_PSR471)	16	R/W	8000h	23.5.16/ 906
410	INTC Priority Select Register (INTC_PSR472)	16	R/W	8000h	23.5.16/ 906
412	INTC Priority Select Register (INTC_PSR473)	16	R/W	8000h	23.5.16/ 906
414	INTC Priority Select Register (INTC_PSR474)	16	R/W	8000h	23.5.16/ 906
416	INTC Priority Select Register (INTC_PSR475)	16	R/W	8000h	23.5.16/ 906
418	INTC Priority Select Register (INTC_PSR476)	16	R/W	8000h	23.5.16/ 906
41A	INTC Priority Select Register (INTC_PSR477)	16	R/W	8000h	23.5.16/ 906
41C	INTC Priority Select Register (INTC_PSR478)	16	R/W	8000h	23.5.16/ 906
41E	INTC Priority Select Register (INTC_PSR479)	16	R/W	8000h	23.5.16/ 906
420	INTC Priority Select Register (INTC_PSR480)	16	R/W	8000h	23.5.16/ 906
422	INTC Priority Select Register (INTC_PSR481)	16	R/W	8000h	23.5.16/ 906
424	INTC Priority Select Register (INTC_PSR482)	16	R/W	8000h	23.5.16/ 906
426	INTC Priority Select Register (INTC_PSR483)	16	R/W	8000h	23.5.16/ 906
428	INTC Priority Select Register (INTC_PSR484)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
42A	INTC Priority Select Register (INTC_PSR485)	16	R/W	8000h	23.5.16/ 906
42C	INTC Priority Select Register (INTC_PSR486)	16	R/W	8000h	23.5.16/ 906
42E	INTC Priority Select Register (INTC_PSR487)	16	R/W	8000h	23.5.16/ 906
430	INTC Priority Select Register (INTC_PSR488)	16	R/W	8000h	23.5.16/ 906
432	INTC Priority Select Register (INTC_PSR489)	16	R/W	8000h	23.5.16/ 906
434	INTC Priority Select Register (INTC_PSR490)	16	R/W	8000h	23.5.16/ 906
436	INTC Priority Select Register (INTC_PSR491)	16	R/W	8000h	23.5.16/ 906
438	INTC Priority Select Register (INTC_PSR492)	16	R/W	8000h	23.5.16/ 906
43A	INTC Priority Select Register (INTC_PSR493)	16	R/W	8000h	23.5.16/ 906
43C	INTC Priority Select Register (INTC_PSR494)	16	R/W	8000h	23.5.16/ 906
43E	INTC Priority Select Register (INTC_PSR495)	16	R/W	8000h	23.5.16/ 906
440	INTC Priority Select Register (INTC_PSR496)	16	R/W	8000h	23.5.16/ 906
442	INTC Priority Select Register (INTC_PSR497)	16	R/W	8000h	23.5.16/ 906
444	INTC Priority Select Register (INTC_PSR498)	16	R/W	8000h	23.5.16/ 906
446	INTC Priority Select Register (INTC_PSR499)	16	R/W	8000h	23.5.16/ 906
448	INTC Priority Select Register (INTC_PSR500)	16	R/W	8000h	23.5.16/ 906
44A	INTC Priority Select Register (INTC_PSR501)	16	R/W	8000h	23.5.16/ 906
44C	INTC Priority Select Register (INTC_PSR502)	16	R/W	8000h	23.5.16/ 906
44E	INTC Priority Select Register (INTC_PSR503)	16	R/W	8000h	23.5.16/ 906
450	INTC Priority Select Register (INTC_PSR504)	16	R/W	8000h	23.5.16/ 906
452	INTC Priority Select Register (INTC_PSR505)	16	R/W	8000h	23.5.16/ 906
454	INTC Priority Select Register (INTC_PSR506)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
456	INTC Priority Select Register (INTC_PSR507)	16	R/W	8000h	23.5.16/ 906
458	INTC Priority Select Register (INTC_PSR508)	16	R/W	8000h	23.5.16/ 906
45A	INTC Priority Select Register (INTC_PSR509)	16	R/W	8000h	23.5.16/ 906
45C	INTC Priority Select Register (INTC_PSR510)	16	R/W	8000h	23.5.16/ 906
45E	INTC Priority Select Register (INTC_PSR511)	16	R/W	8000h	23.5.16/ 906
460	INTC Priority Select Register (INTC_PSR512)	16	R/W	8000h	23.5.16/ 906
462	INTC Priority Select Register (INTC_PSR513)	16	R/W	8000h	23.5.16/ 906
464	INTC Priority Select Register (INTC_PSR514)	16	R/W	8000h	23.5.16/ 906
466	INTC Priority Select Register (INTC_PSR515)	16	R/W	8000h	23.5.16/ 906
468	INTC Priority Select Register (INTC_PSR516)	16	R/W	8000h	23.5.16/ 906
46A	INTC Priority Select Register (INTC_PSR517)	16	R/W	8000h	23.5.16/ 906
46C	INTC Priority Select Register (INTC_PSR518)	16	R/W	8000h	23.5.16/ 906
46E	INTC Priority Select Register (INTC_PSR519)	16	R/W	8000h	23.5.16/ 906
470	INTC Priority Select Register (INTC_PSR520)	16	R/W	8000h	23.5.16/ 906
472	INTC Priority Select Register (INTC_PSR521)	16	R/W	8000h	23.5.16/ 906
474	INTC Priority Select Register (INTC_PSR522)	16	R/W	8000h	23.5.16/ 906
476	INTC Priority Select Register (INTC_PSR523)	16	R/W	8000h	23.5.16/ 906
478	INTC Priority Select Register (INTC_PSR524)	16	R/W	8000h	23.5.16/ 906
47A	INTC Priority Select Register (INTC_PSR525)	16	R/W	8000h	23.5.16/ 906
47C	INTC Priority Select Register (INTC_PSR526)	16	R/W	8000h	23.5.16/ 906
47E	INTC Priority Select Register (INTC_PSR527)	16	R/W	8000h	23.5.16/ 906
480	INTC Priority Select Register (INTC_PSR528)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
482	INTC Priority Select Register (INTC_PSR529)	16	R/W	8000h	23.5.16/ 906
484	INTC Priority Select Register (INTC_PSR530)	16	R/W	8000h	23.5.16/ 906
486	INTC Priority Select Register (INTC_PSR531)	16	R/W	8000h	23.5.16/ 906
488	INTC Priority Select Register (INTC_PSR532)	16	R/W	8000h	23.5.16/ 906
48A	INTC Priority Select Register (INTC_PSR533)	16	R/W	8000h	23.5.16/ 906
48C	INTC Priority Select Register (INTC_PSR534)	16	R/W	8000h	23.5.16/ 906
48E	INTC Priority Select Register (INTC_PSR535)	16	R/W	8000h	23.5.16/ 906
490	INTC Priority Select Register (INTC_PSR536)	16	R/W	8000h	23.5.16/ 906
492	INTC Priority Select Register (INTC_PSR537)	16	R/W	8000h	23.5.16/ 906
494	INTC Priority Select Register (INTC_PSR538)	16	R/W	8000h	23.5.16/ 906
496	INTC Priority Select Register (INTC_PSR539)	16	R/W	8000h	23.5.16/ 906
498	INTC Priority Select Register (INTC_PSR540)	16	R/W	8000h	23.5.16/ 906
49A	INTC Priority Select Register (INTC_PSR541)	16	R/W	8000h	23.5.16/ 906
49C	INTC Priority Select Register (INTC_PSR542)	16	R/W	8000h	23.5.16/ 906
49E	INTC Priority Select Register (INTC_PSR543)	16	R/W	8000h	23.5.16/ 906
4A0	INTC Priority Select Register (INTC_PSR544)	16	R/W	8000h	23.5.16/ 906
4A2	INTC Priority Select Register (INTC_PSR545)	16	R/W	8000h	23.5.16/ 906
4A4	INTC Priority Select Register (INTC_PSR546)	16	R/W	8000h	23.5.16/ 906
4A6	INTC Priority Select Register (INTC_PSR547)	16	R/W	8000h	23.5.16/ 906
4A8	INTC Priority Select Register (INTC_PSR548)	16	R/W	8000h	23.5.16/ 906
4AA	INTC Priority Select Register (INTC_PSR549)	16	R/W	8000h	23.5.16/ 906
4AC	INTC Priority Select Register (INTC_PSR550)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4AE	INTC Priority Select Register (INTC_PSR551)	16	R/W	8000h	23.5.16/ 906
4B0	INTC Priority Select Register (INTC_PSR552)	16	R/W	8000h	23.5.16/ 906
4B2	INTC Priority Select Register (INTC_PSR553)	16	R/W	8000h	23.5.16/ 906
4B4	INTC Priority Select Register (INTC_PSR554)	16	R/W	8000h	23.5.16/ 906
4B6	INTC Priority Select Register (INTC_PSR555)	16	R/W	8000h	23.5.16/ 906
4B8	INTC Priority Select Register (INTC_PSR556)	16	R/W	8000h	23.5.16/ 906
4BA	INTC Priority Select Register (INTC_PSR557)	16	R/W	8000h	23.5.16/ 906
4BC	INTC Priority Select Register (INTC_PSR558)	16	R/W	8000h	23.5.16/ 906
4BE	INTC Priority Select Register (INTC_PSR559)	16	R/W	8000h	23.5.16/ 906
4C0	INTC Priority Select Register (INTC_PSR560)	16	R/W	8000h	23.5.16/ 906
4C2	INTC Priority Select Register (INTC_PSR561)	16	R/W	8000h	23.5.16/ 906
4C4	INTC Priority Select Register (INTC_PSR562)	16	R/W	8000h	23.5.16/ 906
4C6	INTC Priority Select Register (INTC_PSR563)	16	R/W	8000h	23.5.16/ 906
4C8	INTC Priority Select Register (INTC_PSR564)	16	R/W	8000h	23.5.16/ 906
4CA	INTC Priority Select Register (INTC_PSR565)	16	R/W	8000h	23.5.16/ 906
4CC	INTC Priority Select Register (INTC_PSR566)	16	R/W	8000h	23.5.16/ 906
4CE	INTC Priority Select Register (INTC_PSR567)	16	R/W	8000h	23.5.16/ 906
4D0	INTC Priority Select Register (INTC_PSR568)	16	R/W	8000h	23.5.16/ 906
4D2	INTC Priority Select Register (INTC_PSR569)	16	R/W	8000h	23.5.16/ 906
4D4	INTC Priority Select Register (INTC_PSR570)	16	R/W	8000h	23.5.16/ 906
4D6	INTC Priority Select Register (INTC_PSR571)	16	R/W	8000h	23.5.16/ 906
4D8	INTC Priority Select Register (INTC_PSR572)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4DA	INTC Priority Select Register (INTC_PSR573)	16	R/W	8000h	23.5.16/ 906
4DC	INTC Priority Select Register (INTC_PSR574)	16	R/W	8000h	23.5.16/ 906
4DE	INTC Priority Select Register (INTC_PSR575)	16	R/W	8000h	23.5.16/ 906
4E0	INTC Priority Select Register (INTC_PSR576)	16	R/W	8000h	23.5.16/ 906
4E2	INTC Priority Select Register (INTC_PSR577)	16	R/W	8000h	23.5.16/ 906
4E4	INTC Priority Select Register (INTC_PSR578)	16	R/W	8000h	23.5.16/ 906
4E6	INTC Priority Select Register (INTC_PSR579)	16	R/W	8000h	23.5.16/ 906
4E8	INTC Priority Select Register (INTC_PSR580)	16	R/W	8000h	23.5.16/ 906
4EA	INTC Priority Select Register (INTC_PSR581)	16	R/W	8000h	23.5.16/ 906
4EC	INTC Priority Select Register (INTC_PSR582)	16	R/W	8000h	23.5.16/ 906
4EE	INTC Priority Select Register (INTC_PSR583)	16	R/W	8000h	23.5.16/ 906
4F0	INTC Priority Select Register (INTC_PSR584)	16	R/W	8000h	23.5.16/ 906
4F2	INTC Priority Select Register (INTC_PSR585)	16	R/W	8000h	23.5.16/ 906
4F4	INTC Priority Select Register (INTC_PSR586)	16	R/W	8000h	23.5.16/ 906
4F6	INTC Priority Select Register (INTC_PSR587)	16	R/W	8000h	23.5.16/ 906
4F8	INTC Priority Select Register (INTC_PSR588)	16	R/W	8000h	23.5.16/ 906
4FA	INTC Priority Select Register (INTC_PSR589)	16	R/W	8000h	23.5.16/ 906
4FC	INTC Priority Select Register (INTC_PSR590)	16	R/W	8000h	23.5.16/ 906
4FE	INTC Priority Select Register (INTC_PSR591)	16	R/W	8000h	23.5.16/ 906
500	INTC Priority Select Register (INTC_PSR592)	16	R/W	8000h	23.5.16/ 906
502	INTC Priority Select Register (INTC_PSR593)	16	R/W	8000h	23.5.16/ 906
504	INTC Priority Select Register (INTC_PSR594)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
506	INTC Priority Select Register (INTC_PSR595)	16	R/W	8000h	23.5.16/ 906
508	INTC Priority Select Register (INTC_PSR596)	16	R/W	8000h	23.5.16/ 906
50A	INTC Priority Select Register (INTC_PSR597)	16	R/W	8000h	23.5.16/ 906
50C	INTC Priority Select Register (INTC_PSR598)	16	R/W	8000h	23.5.16/ 906
50E	INTC Priority Select Register (INTC_PSR599)	16	R/W	8000h	23.5.16/ 906
510	INTC Priority Select Register (INTC_PSR600)	16	R/W	8000h	23.5.16/ 906
512	INTC Priority Select Register (INTC_PSR601)	16	R/W	8000h	23.5.16/ 906
514	INTC Priority Select Register (INTC_PSR602)	16	R/W	8000h	23.5.16/ 906
516	INTC Priority Select Register (INTC_PSR603)	16	R/W	8000h	23.5.16/ 906
518	INTC Priority Select Register (INTC_PSR604)	16	R/W	8000h	23.5.16/ 906
51A	INTC Priority Select Register (INTC_PSR605)	16	R/W	8000h	23.5.16/ 906
51C	INTC Priority Select Register (INTC_PSR606)	16	R/W	8000h	23.5.16/ 906
51E	INTC Priority Select Register (INTC_PSR607)	16	R/W	8000h	23.5.16/ 906
520	INTC Priority Select Register (INTC_PSR608)	16	R/W	8000h	23.5.16/ 906
522	INTC Priority Select Register (INTC_PSR609)	16	R/W	8000h	23.5.16/ 906
524	INTC Priority Select Register (INTC_PSR610)	16	R/W	8000h	23.5.16/ 906
526	INTC Priority Select Register (INTC_PSR611)	16	R/W	8000h	23.5.16/ 906
528	INTC Priority Select Register (INTC_PSR612)	16	R/W	8000h	23.5.16/ 906
52A	INTC Priority Select Register (INTC_PSR613)	16	R/W	8000h	23.5.16/ 906
52C	INTC Priority Select Register (INTC_PSR614)	16	R/W	8000h	23.5.16/ 906
52E	INTC Priority Select Register (INTC_PSR615)	16	R/W	8000h	23.5.16/ 906
530	INTC Priority Select Register (INTC_PSR616)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
532	INTC Priority Select Register (INTC_PSR617)	16	R/W	8000h	23.5.16/ 906
534	INTC Priority Select Register (INTC_PSR618)	16	R/W	8000h	23.5.16/ 906
536	INTC Priority Select Register (INTC_PSR619)	16	R/W	8000h	23.5.16/ 906
538	INTC Priority Select Register (INTC_PSR620)	16	R/W	8000h	23.5.16/ 906
53A	INTC Priority Select Register (INTC_PSR621)	16	R/W	8000h	23.5.16/ 906
53C	INTC Priority Select Register (INTC_PSR622)	16	R/W	8000h	23.5.16/ 906
53E	INTC Priority Select Register (INTC_PSR623)	16	R/W	8000h	23.5.16/ 906
540	INTC Priority Select Register (INTC_PSR624)	16	R/W	8000h	23.5.16/ 906
542	INTC Priority Select Register (INTC_PSR625)	16	R/W	8000h	23.5.16/ 906
544	INTC Priority Select Register (INTC_PSR626)	16	R/W	8000h	23.5.16/ 906
546	INTC Priority Select Register (INTC_PSR627)	16	R/W	8000h	23.5.16/ 906
548	INTC Priority Select Register (INTC_PSR628)	16	R/W	8000h	23.5.16/ 906
54A	INTC Priority Select Register (INTC_PSR629)	16	R/W	8000h	23.5.16/ 906
54C	INTC Priority Select Register (INTC_PSR630)	16	R/W	8000h	23.5.16/ 906
54E	INTC Priority Select Register (INTC_PSR631)	16	R/W	8000h	23.5.16/ 906
550	INTC Priority Select Register (INTC_PSR632)	16	R/W	8000h	23.5.16/ 906
552	INTC Priority Select Register (INTC_PSR633)	16	R/W	8000h	23.5.16/ 906
554	INTC Priority Select Register (INTC_PSR634)	16	R/W	8000h	23.5.16/ 906
556	INTC Priority Select Register (INTC_PSR635)	16	R/W	8000h	23.5.16/ 906
558	INTC Priority Select Register (INTC_PSR636)	16	R/W	8000h	23.5.16/ 906
55A	INTC Priority Select Register (INTC_PSR637)	16	R/W	8000h	23.5.16/ 906
55C	INTC Priority Select Register (INTC_PSR638)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
55E	INTC Priority Select Register (INTC_PSR639)	16	R/W	8000h	23.5.16/ 906
560	INTC Priority Select Register (INTC_PSR640)	16	R/W	8000h	23.5.16/ 906
562	INTC Priority Select Register (INTC_PSR641)	16	R/W	8000h	23.5.16/ 906
564	INTC Priority Select Register (INTC_PSR642)	16	R/W	8000h	23.5.16/ 906
566	INTC Priority Select Register (INTC_PSR643)	16	R/W	8000h	23.5.16/ 906
568	INTC Priority Select Register (INTC_PSR644)	16	R/W	8000h	23.5.16/ 906
56A	INTC Priority Select Register (INTC_PSR645)	16	R/W	8000h	23.5.16/ 906
56C	INTC Priority Select Register (INTC_PSR646)	16	R/W	8000h	23.5.16/ 906
56E	INTC Priority Select Register (INTC_PSR647)	16	R/W	8000h	23.5.16/ 906
570	INTC Priority Select Register (INTC_PSR648)	16	R/W	8000h	23.5.16/ 906
572	INTC Priority Select Register (INTC_PSR649)	16	R/W	8000h	23.5.16/ 906
574	INTC Priority Select Register (INTC_PSR650)	16	R/W	8000h	23.5.16/ 906
576	INTC Priority Select Register (INTC_PSR651)	16	R/W	8000h	23.5.16/ 906
578	INTC Priority Select Register (INTC_PSR652)	16	R/W	8000h	23.5.16/ 906
57A	INTC Priority Select Register (INTC_PSR653)	16	R/W	8000h	23.5.16/ 906
57C	INTC Priority Select Register (INTC_PSR654)	16	R/W	8000h	23.5.16/ 906
57E	INTC Priority Select Register (INTC_PSR655)	16	R/W	8000h	23.5.16/ 906
580	INTC Priority Select Register (INTC_PSR656)	16	R/W	8000h	23.5.16/ 906
582	INTC Priority Select Register (INTC_PSR657)	16	R/W	8000h	23.5.16/ 906
584	INTC Priority Select Register (INTC_PSR658)	16	R/W	8000h	23.5.16/ 906
586	INTC Priority Select Register (INTC_PSR659)	16	R/W	8000h	23.5.16/ 906
588	INTC Priority Select Register (INTC_PSR660)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
58A	INTC Priority Select Register (INTC_PSR661)	16	R/W	8000h	23.5.16/ 906
58C	INTC Priority Select Register (INTC_PSR662)	16	R/W	8000h	23.5.16/ 906
58E	INTC Priority Select Register (INTC_PSR663)	16	R/W	8000h	23.5.16/ 906
590	INTC Priority Select Register (INTC_PSR664)	16	R/W	8000h	23.5.16/ 906
592	INTC Priority Select Register (INTC_PSR665)	16	R/W	8000h	23.5.16/ 906
594	INTC Priority Select Register (INTC_PSR666)	16	R/W	8000h	23.5.16/ 906
596	INTC Priority Select Register (INTC_PSR667)	16	R/W	8000h	23.5.16/ 906
598	INTC Priority Select Register (INTC_PSR668)	16	R/W	8000h	23.5.16/ 906
59A	INTC Priority Select Register (INTC_PSR669)	16	R/W	8000h	23.5.16/ 906
59C	INTC Priority Select Register (INTC_PSR670)	16	R/W	8000h	23.5.16/ 906
59E	INTC Priority Select Register (INTC_PSR671)	16	R/W	8000h	23.5.16/ 906
5A0	INTC Priority Select Register (INTC_PSR672)	16	R/W	8000h	23.5.16/ 906
5A2	INTC Priority Select Register (INTC_PSR673)	16	R/W	8000h	23.5.16/ 906
5A4	INTC Priority Select Register (INTC_PSR674)	16	R/W	8000h	23.5.16/ 906
5A6	INTC Priority Select Register (INTC_PSR675)	16	R/W	8000h	23.5.16/ 906
5A8	INTC Priority Select Register (INTC_PSR676)	16	R/W	8000h	23.5.16/ 906
5AA	INTC Priority Select Register (INTC_PSR677)	16	R/W	8000h	23.5.16/ 906
5AC	INTC Priority Select Register (INTC_PSR678)	16	R/W	8000h	23.5.16/ 906
5AE	INTC Priority Select Register (INTC_PSR679)	16	R/W	8000h	23.5.16/ 906
5B0	INTC Priority Select Register (INTC_PSR680)	16	R/W	8000h	23.5.16/ 906
5B2	INTC Priority Select Register (INTC_PSR681)	16	R/W	8000h	23.5.16/ 906
5B4	INTC Priority Select Register (INTC_PSR682)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5B6	INTC Priority Select Register (INTC_PSR683)	16	R/W	8000h	23.5.16/ 906
5B8	INTC Priority Select Register (INTC_PSR684)	16	R/W	8000h	23.5.16/ 906
5BA	INTC Priority Select Register (INTC_PSR685)	16	R/W	8000h	23.5.16/ 906
5BC	INTC Priority Select Register (INTC_PSR686)	16	R/W	8000h	23.5.16/ 906
5BE	INTC Priority Select Register (INTC_PSR687)	16	R/W	8000h	23.5.16/ 906
5C0	INTC Priority Select Register (INTC_PSR688)	16	R/W	8000h	23.5.16/ 906
5C2	INTC Priority Select Register (INTC_PSR689)	16	R/W	8000h	23.5.16/ 906
5C4	INTC Priority Select Register (INTC_PSR690)	16	R/W	8000h	23.5.16/ 906
5C6	INTC Priority Select Register (INTC_PSR691)	16	R/W	8000h	23.5.16/ 906
5C8	INTC Priority Select Register (INTC_PSR692)	16	R/W	8000h	23.5.16/ 906
5CA	INTC Priority Select Register (INTC_PSR693)	16	R/W	8000h	23.5.16/ 906
5CC	INTC Priority Select Register (INTC_PSR694)	16	R/W	8000h	23.5.16/ 906
5CE	INTC Priority Select Register (INTC_PSR695)	16	R/W	8000h	23.5.16/ 906
5D0	INTC Priority Select Register (INTC_PSR696)	16	R/W	8000h	23.5.16/ 906
5D2	INTC Priority Select Register (INTC_PSR697)	16	R/W	8000h	23.5.16/ 906
5D4	INTC Priority Select Register (INTC_PSR698)	16	R/W	8000h	23.5.16/ 906
5D6	INTC Priority Select Register (INTC_PSR699)	16	R/W	8000h	23.5.16/ 906
5D8	INTC Priority Select Register (INTC_PSR700)	16	R/W	8000h	23.5.16/ 906
5DA	INTC Priority Select Register (INTC_PSR701)	16	R/W	8000h	23.5.16/ 906
5DC	INTC Priority Select Register (INTC_PSR702)	16	R/W	8000h	23.5.16/ 906
5DE	INTC Priority Select Register (INTC_PSR703)	16	R/W	8000h	23.5.16/ 906
5E0	INTC Priority Select Register (INTC_PSR704)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5E2	INTC Priority Select Register (INTC_PSR705)	16	R/W	8000h	23.5.16/ 906
5E4	INTC Priority Select Register (INTC_PSR706)	16	R/W	8000h	23.5.16/ 906
5E6	INTC Priority Select Register (INTC_PSR707)	16	R/W	8000h	23.5.16/ 906
5E8	INTC Priority Select Register (INTC_PSR708)	16	R/W	8000h	23.5.16/ 906
5EA	INTC Priority Select Register (INTC_PSR709)	16	R/W	8000h	23.5.16/ 906
5EC	INTC Priority Select Register (INTC_PSR710)	16	R/W	8000h	23.5.16/ 906
5EE	INTC Priority Select Register (INTC_PSR711)	16	R/W	8000h	23.5.16/ 906
5F0	INTC Priority Select Register (INTC_PSR712)	16	R/W	8000h	23.5.16/ 906
5F2	INTC Priority Select Register (INTC_PSR713)	16	R/W	8000h	23.5.16/ 906
5F4	INTC Priority Select Register (INTC_PSR714)	16	R/W	8000h	23.5.16/ 906
5F6	INTC Priority Select Register (INTC_PSR715)	16	R/W	8000h	23.5.16/ 906
5F8	INTC Priority Select Register (INTC_PSR716)	16	R/W	8000h	23.5.16/ 906
5FA	INTC Priority Select Register (INTC_PSR717)	16	R/W	8000h	23.5.16/ 906
5FC	INTC Priority Select Register (INTC_PSR718)	16	R/W	8000h	23.5.16/ 906
5FE	INTC Priority Select Register (INTC_PSR719)	16	R/W	8000h	23.5.16/ 906
600	INTC Priority Select Register (INTC_PSR720)	16	R/W	8000h	23.5.16/ 906
602	INTC Priority Select Register (INTC_PSR721)	16	R/W	8000h	23.5.16/ 906
604	INTC Priority Select Register (INTC_PSR722)	16	R/W	8000h	23.5.16/ 906
606	INTC Priority Select Register (INTC_PSR723)	16	R/W	8000h	23.5.16/ 906
608	INTC Priority Select Register (INTC_PSR724)	16	R/W	8000h	23.5.16/ 906
60A	INTC Priority Select Register (INTC_PSR725)	16	R/W	8000h	23.5.16/ 906
60C	INTC Priority Select Register (INTC_PSR726)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60E	INTC Priority Select Register (INTC_PSR727)	16	R/W	8000h	23.5.16/ 906
610	INTC Priority Select Register (INTC_PSR728)	16	R/W	8000h	23.5.16/ 906
612	INTC Priority Select Register (INTC_PSR729)	16	R/W	8000h	23.5.16/ 906
614	INTC Priority Select Register (INTC_PSR730)	16	R/W	8000h	23.5.16/ 906
616	INTC Priority Select Register (INTC_PSR731)	16	R/W	8000h	23.5.16/ 906
618	INTC Priority Select Register (INTC_PSR732)	16	R/W	8000h	23.5.16/ 906
61A	INTC Priority Select Register (INTC_PSR733)	16	R/W	8000h	23.5.16/ 906
61C	INTC Priority Select Register (INTC_PSR734)	16	R/W	8000h	23.5.16/ 906
61E	INTC Priority Select Register (INTC_PSR735)	16	R/W	8000h	23.5.16/ 906
620	INTC Priority Select Register (INTC_PSR736)	16	R/W	8000h	23.5.16/ 906
622	INTC Priority Select Register (INTC_PSR737)	16	R/W	8000h	23.5.16/ 906
624	INTC Priority Select Register (INTC_PSR738)	16	R/W	8000h	23.5.16/ 906
626	INTC Priority Select Register (INTC_PSR739)	16	R/W	8000h	23.5.16/ 906
628	INTC Priority Select Register (INTC_PSR740)	16	R/W	8000h	23.5.16/ 906
62A	INTC Priority Select Register (INTC_PSR741)	16	R/W	8000h	23.5.16/ 906
62C	INTC Priority Select Register (INTC_PSR742)	16	R/W	8000h	23.5.16/ 906
62E	INTC Priority Select Register (INTC_PSR743)	16	R/W	8000h	23.5.16/ 906
630	INTC Priority Select Register (INTC_PSR744)	16	R/W	8000h	23.5.16/ 906
632	INTC Priority Select Register (INTC_PSR745)	16	R/W	8000h	23.5.16/ 906
634	INTC Priority Select Register (INTC_PSR746)	16	R/W	8000h	23.5.16/ 906
636	INTC Priority Select Register (INTC_PSR747)	16	R/W	8000h	23.5.16/ 906
638	INTC Priority Select Register (INTC_PSR748)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63A	INTC Priority Select Register (INTC_PSR749)	16	R/W	8000h	23.5.16/ 906
63C	INTC Priority Select Register (INTC_PSR750)	16	R/W	8000h	23.5.16/ 906
63E	INTC Priority Select Register (INTC_PSR751)	16	R/W	8000h	23.5.16/ 906
640	INTC Priority Select Register (INTC_PSR752)	16	R/W	8000h	23.5.16/ 906
642	INTC Priority Select Register (INTC_PSR753)	16	R/W	8000h	23.5.16/ 906
644	INTC Priority Select Register (INTC_PSR754)	16	R/W	8000h	23.5.16/ 906
646	INTC Priority Select Register (INTC_PSR755)	16	R/W	8000h	23.5.16/ 906
648	INTC Priority Select Register (INTC_PSR756)	16	R/W	8000h	23.5.16/ 906
64A	INTC Priority Select Register (INTC_PSR757)	16	R/W	8000h	23.5.16/ 906
64C	INTC Priority Select Register (INTC_PSR758)	16	R/W	8000h	23.5.16/ 906
64E	INTC Priority Select Register (INTC_PSR759)	16	R/W	8000h	23.5.16/ 906
650	INTC Priority Select Register (INTC_PSR760)	16	R/W	8000h	23.5.16/ 906
652	INTC Priority Select Register (INTC_PSR761)	16	R/W	8000h	23.5.16/ 906
654	INTC Priority Select Register (INTC_PSR762)	16	R/W	8000h	23.5.16/ 906
656	INTC Priority Select Register (INTC_PSR763)	16	R/W	8000h	23.5.16/ 906
658	INTC Priority Select Register (INTC_PSR764)	16	R/W	8000h	23.5.16/ 906
65A	INTC Priority Select Register (INTC_PSR765)	16	R/W	8000h	23.5.16/ 906
65C	INTC Priority Select Register (INTC_PSR766)	16	R/W	8000h	23.5.16/ 906
65E	INTC Priority Select Register (INTC_PSR767)	16	R/W	8000h	23.5.16/ 906
660	INTC Priority Select Register (INTC_PSR768)	16	R/W	8000h	23.5.16/ 906
662	INTC Priority Select Register (INTC_PSR769)	16	R/W	8000h	23.5.16/ 906
664	INTC Priority Select Register (INTC_PSR770)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
666	INTC Priority Select Register (INTC_PSR771)	16	R/W	8000h	23.5.16/ 906
668	INTC Priority Select Register (INTC_PSR772)	16	R/W	8000h	23.5.16/ 906
66A	INTC Priority Select Register (INTC_PSR773)	16	R/W	8000h	23.5.16/ 906
66C	INTC Priority Select Register (INTC_PSR774)	16	R/W	8000h	23.5.16/ 906
66E	INTC Priority Select Register (INTC_PSR775)	16	R/W	8000h	23.5.16/ 906
670	INTC Priority Select Register (INTC_PSR776)	16	R/W	8000h	23.5.16/ 906
672	INTC Priority Select Register (INTC_PSR777)	16	R/W	8000h	23.5.16/ 906
674	INTC Priority Select Register (INTC_PSR778)	16	R/W	8000h	23.5.16/ 906
676	INTC Priority Select Register (INTC_PSR779)	16	R/W	8000h	23.5.16/ 906
678	INTC Priority Select Register (INTC_PSR780)	16	R/W	8000h	23.5.16/ 906
67A	INTC Priority Select Register (INTC_PSR781)	16	R/W	8000h	23.5.16/ 906
67C	INTC Priority Select Register (INTC_PSR782)	16	R/W	8000h	23.5.16/ 906
67E	INTC Priority Select Register (INTC_PSR783)	16	R/W	8000h	23.5.16/ 906
680	INTC Priority Select Register (INTC_PSR784)	16	R/W	8000h	23.5.16/ 906
682	INTC Priority Select Register (INTC_PSR785)	16	R/W	8000h	23.5.16/ 906
684	INTC Priority Select Register (INTC_PSR786)	16	R/W	8000h	23.5.16/ 906
686	INTC Priority Select Register (INTC_PSR787)	16	R/W	8000h	23.5.16/ 906
688	INTC Priority Select Register (INTC_PSR788)	16	R/W	8000h	23.5.16/ 906
68A	INTC Priority Select Register (INTC_PSR789)	16	R/W	8000h	23.5.16/ 906
68C	INTC Priority Select Register (INTC_PSR790)	16	R/W	8000h	23.5.16/ 906
68E	INTC Priority Select Register (INTC_PSR791)	16	R/W	8000h	23.5.16/ 906
690	INTC Priority Select Register (INTC_PSR792)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
692	INTC Priority Select Register (INTC_PSR793)	16	R/W	8000h	23.5.16/ 906
694	INTC Priority Select Register (INTC_PSR794)	16	R/W	8000h	23.5.16/ 906
696	INTC Priority Select Register (INTC_PSR795)	16	R/W	8000h	23.5.16/ 906
698	INTC Priority Select Register (INTC_PSR796)	16	R/W	8000h	23.5.16/ 906
69A	INTC Priority Select Register (INTC_PSR797)	16	R/W	8000h	23.5.16/ 906
69C	INTC Priority Select Register (INTC_PSR798)	16	R/W	8000h	23.5.16/ 906
69E	INTC Priority Select Register (INTC_PSR799)	16	R/W	8000h	23.5.16/ 906
6A0	INTC Priority Select Register (INTC_PSR800)	16	R/W	8000h	23.5.16/ 906
6A2	INTC Priority Select Register (INTC_PSR801)	16	R/W	8000h	23.5.16/ 906
6A4	INTC Priority Select Register (INTC_PSR802)	16	R/W	8000h	23.5.16/ 906
6A6	INTC Priority Select Register (INTC_PSR803)	16	R/W	8000h	23.5.16/ 906
6A8	INTC Priority Select Register (INTC_PSR804)	16	R/W	8000h	23.5.16/ 906
6AA	INTC Priority Select Register (INTC_PSR805)	16	R/W	8000h	23.5.16/ 906
6AC	INTC Priority Select Register (INTC_PSR806)	16	R/W	8000h	23.5.16/ 906
6AE	INTC Priority Select Register (INTC_PSR807)	16	R/W	8000h	23.5.16/ 906
6B0	INTC Priority Select Register (INTC_PSR808)	16	R/W	8000h	23.5.16/ 906
6B2	INTC Priority Select Register (INTC_PSR809)	16	R/W	8000h	23.5.16/ 906
6B4	INTC Priority Select Register (INTC_PSR810)	16	R/W	8000h	23.5.16/ 906
6B6	INTC Priority Select Register (INTC_PSR811)	16	R/W	8000h	23.5.16/ 906
6B8	INTC Priority Select Register (INTC_PSR812)	16	R/W	8000h	23.5.16/ 906
6BA	INTC Priority Select Register (INTC_PSR813)	16	R/W	8000h	23.5.16/ 906
6BC	INTC Priority Select Register (INTC_PSR814)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6BE	INTC Priority Select Register (INTC_PSR815)	16	R/W	8000h	23.5.16/ 906
6C0	INTC Priority Select Register (INTC_PSR816)	16	R/W	8000h	23.5.16/ 906
6C2	INTC Priority Select Register (INTC_PSR817)	16	R/W	8000h	23.5.16/ 906
6C4	INTC Priority Select Register (INTC_PSR818)	16	R/W	8000h	23.5.16/ 906
6C6	INTC Priority Select Register (INTC_PSR819)	16	R/W	8000h	23.5.16/ 906
6C8	INTC Priority Select Register (INTC_PSR820)	16	R/W	8000h	23.5.16/ 906
6CA	INTC Priority Select Register (INTC_PSR821)	16	R/W	8000h	23.5.16/ 906
6CC	INTC Priority Select Register (INTC_PSR822)	16	R/W	8000h	23.5.16/ 906
6CE	INTC Priority Select Register (INTC_PSR823)	16	R/W	8000h	23.5.16/ 906
6D0	INTC Priority Select Register (INTC_PSR824)	16	R/W	8000h	23.5.16/ 906
6D2	INTC Priority Select Register (INTC_PSR825)	16	R/W	8000h	23.5.16/ 906
6D4	INTC Priority Select Register (INTC_PSR826)	16	R/W	8000h	23.5.16/ 906
6D6	INTC Priority Select Register (INTC_PSR827)	16	R/W	8000h	23.5.16/ 906
6D8	INTC Priority Select Register (INTC_PSR828)	16	R/W	8000h	23.5.16/ 906
6DA	INTC Priority Select Register (INTC_PSR829)	16	R/W	8000h	23.5.16/ 906
6DC	INTC Priority Select Register (INTC_PSR830)	16	R/W	8000h	23.5.16/ 906
6DE	INTC Priority Select Register (INTC_PSR831)	16	R/W	8000h	23.5.16/ 906
6E0	INTC Priority Select Register (INTC_PSR832)	16	R/W	8000h	23.5.16/ 906
6E2	INTC Priority Select Register (INTC_PSR833)	16	R/W	8000h	23.5.16/ 906
6E4	INTC Priority Select Register (INTC_PSR834)	16	R/W	8000h	23.5.16/ 906
6E6	INTC Priority Select Register (INTC_PSR835)	16	R/W	8000h	23.5.16/ 906
6E8	INTC Priority Select Register (INTC_PSR836)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6EA	INTC Priority Select Register (INTC_PSR837)	16	R/W	8000h	23.5.16/ 906
6EC	INTC Priority Select Register (INTC_PSR838)	16	R/W	8000h	23.5.16/ 906
6EE	INTC Priority Select Register (INTC_PSR839)	16	R/W	8000h	23.5.16/ 906
6F0	INTC Priority Select Register (INTC_PSR840)	16	R/W	8000h	23.5.16/ 906
6F2	INTC Priority Select Register (INTC_PSR841)	16	R/W	8000h	23.5.16/ 906
6F4	INTC Priority Select Register (INTC_PSR842)	16	R/W	8000h	23.5.16/ 906
6F6	INTC Priority Select Register (INTC_PSR843)	16	R/W	8000h	23.5.16/ 906
6F8	INTC Priority Select Register (INTC_PSR844)	16	R/W	8000h	23.5.16/ 906
6FA	INTC Priority Select Register (INTC_PSR845)	16	R/W	8000h	23.5.16/ 906
6FC	INTC Priority Select Register (INTC_PSR846)	16	R/W	8000h	23.5.16/ 906
6FE	INTC Priority Select Register (INTC_PSR847)	16	R/W	8000h	23.5.16/ 906
700	INTC Priority Select Register (INTC_PSR848)	16	R/W	8000h	23.5.16/ 906
702	INTC Priority Select Register (INTC_PSR849)	16	R/W	8000h	23.5.16/ 906
704	INTC Priority Select Register (INTC_PSR850)	16	R/W	8000h	23.5.16/ 906
706	INTC Priority Select Register (INTC_PSR851)	16	R/W	8000h	23.5.16/ 906
708	INTC Priority Select Register (INTC_PSR852)	16	R/W	8000h	23.5.16/ 906
70A	INTC Priority Select Register (INTC_PSR853)	16	R/W	8000h	23.5.16/ 906
70C	INTC Priority Select Register (INTC_PSR854)	16	R/W	8000h	23.5.16/ 906
70E	INTC Priority Select Register (INTC_PSR855)	16	R/W	8000h	23.5.16/ 906
710	INTC Priority Select Register (INTC_PSR856)	16	R/W	8000h	23.5.16/ 906
712	INTC Priority Select Register (INTC_PSR857)	16	R/W	8000h	23.5.16/ 906
714	INTC Priority Select Register (INTC_PSR858)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
716	INTC Priority Select Register (INTC_PSR859)	16	R/W	8000h	23.5.16/ 906
718	INTC Priority Select Register (INTC_PSR860)	16	R/W	8000h	23.5.16/ 906
71A	INTC Priority Select Register (INTC_PSR861)	16	R/W	8000h	23.5.16/ 906
71C	INTC Priority Select Register (INTC_PSR862)	16	R/W	8000h	23.5.16/ 906
71E	INTC Priority Select Register (INTC_PSR863)	16	R/W	8000h	23.5.16/ 906
720	INTC Priority Select Register (INTC_PSR864)	16	R/W	8000h	23.5.16/ 906
722	INTC Priority Select Register (INTC_PSR865)	16	R/W	8000h	23.5.16/ 906
724	INTC Priority Select Register (INTC_PSR866)	16	R/W	8000h	23.5.16/ 906
726	INTC Priority Select Register (INTC_PSR867)	16	R/W	8000h	23.5.16/ 906
728	INTC Priority Select Register (INTC_PSR868)	16	R/W	8000h	23.5.16/ 906
72A	INTC Priority Select Register (INTC_PSR869)	16	R/W	8000h	23.5.16/ 906
72C	INTC Priority Select Register (INTC_PSR870)	16	R/W	8000h	23.5.16/ 906
72E	INTC Priority Select Register (INTC_PSR871)	16	R/W	8000h	23.5.16/ 906
730	INTC Priority Select Register (INTC_PSR872)	16	R/W	8000h	23.5.16/ 906
732	INTC Priority Select Register (INTC_PSR873)	16	R/W	8000h	23.5.16/ 906
734	INTC Priority Select Register (INTC_PSR874)	16	R/W	8000h	23.5.16/ 906
736	INTC Priority Select Register (INTC_PSR875)	16	R/W	8000h	23.5.16/ 906
738	INTC Priority Select Register (INTC_PSR876)	16	R/W	8000h	23.5.16/ 906
73A	INTC Priority Select Register (INTC_PSR877)	16	R/W	8000h	23.5.16/ 906
73C	INTC Priority Select Register (INTC_PSR878)	16	R/W	8000h	23.5.16/ 906
73E	INTC Priority Select Register (INTC_PSR879)	16	R/W	8000h	23.5.16/ 906
740	INTC Priority Select Register (INTC_PSR880)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
742	INTC Priority Select Register (INTC_PSR881)	16	R/W	8000h	23.5.16/ 906
744	INTC Priority Select Register (INTC_PSR882)	16	R/W	8000h	23.5.16/ 906
746	INTC Priority Select Register (INTC_PSR883)	16	R/W	8000h	23.5.16/ 906
748	INTC Priority Select Register (INTC_PSR884)	16	R/W	8000h	23.5.16/ 906
74A	INTC Priority Select Register (INTC_PSR885)	16	R/W	8000h	23.5.16/ 906
74C	INTC Priority Select Register (INTC_PSR886)	16	R/W	8000h	23.5.16/ 906
74E	INTC Priority Select Register (INTC_PSR887)	16	R/W	8000h	23.5.16/ 906
750	INTC Priority Select Register (INTC_PSR888)	16	R/W	8000h	23.5.16/ 906
752	INTC Priority Select Register (INTC_PSR889)	16	R/W	8000h	23.5.16/ 906
754	INTC Priority Select Register (INTC_PSR890)	16	R/W	8000h	23.5.16/ 906
756	INTC Priority Select Register (INTC_PSR891)	16	R/W	8000h	23.5.16/ 906
758	INTC Priority Select Register (INTC_PSR892)	16	R/W	8000h	23.5.16/ 906
75A	INTC Priority Select Register (INTC_PSR893)	16	R/W	8000h	23.5.16/ 906
75C	INTC Priority Select Register (INTC_PSR894)	16	R/W	8000h	23.5.16/ 906
75E	INTC Priority Select Register (INTC_PSR895)	16	R/W	8000h	23.5.16/ 906
760	INTC Priority Select Register (INTC_PSR896)	16	R/W	8000h	23.5.16/ 906
762	INTC Priority Select Register (INTC_PSR897)	16	R/W	8000h	23.5.16/ 906
764	INTC Priority Select Register (INTC_PSR898)	16	R/W	8000h	23.5.16/ 906
766	INTC Priority Select Register (INTC_PSR899)	16	R/W	8000h	23.5.16/ 906
768	INTC Priority Select Register (INTC_PSR900)	16	R/W	8000h	23.5.16/ 906
76A	INTC Priority Select Register (INTC_PSR901)	16	R/W	8000h	23.5.16/ 906
76C	INTC Priority Select Register (INTC_PSR902)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
76E	INTC Priority Select Register (INTC_PSR903)	16	R/W	8000h	23.5.16/ 906
770	INTC Priority Select Register (INTC_PSR904)	16	R/W	8000h	23.5.16/ 906
772	INTC Priority Select Register (INTC_PSR905)	16	R/W	8000h	23.5.16/ 906
774	INTC Priority Select Register (INTC_PSR906)	16	R/W	8000h	23.5.16/ 906
776	INTC Priority Select Register (INTC_PSR907)	16	R/W	8000h	23.5.16/ 906
778	INTC Priority Select Register (INTC_PSR908)	16	R/W	8000h	23.5.16/ 906
77A	INTC Priority Select Register (INTC_PSR909)	16	R/W	8000h	23.5.16/ 906
77C	INTC Priority Select Register (INTC_PSR910)	16	R/W	8000h	23.5.16/ 906
77E	INTC Priority Select Register (INTC_PSR911)	16	R/W	8000h	23.5.16/ 906
780	INTC Priority Select Register (INTC_PSR912)	16	R/W	8000h	23.5.16/ 906
782	INTC Priority Select Register (INTC_PSR913)	16	R/W	8000h	23.5.16/ 906
784	INTC Priority Select Register (INTC_PSR914)	16	R/W	8000h	23.5.16/ 906
786	INTC Priority Select Register (INTC_PSR915)	16	R/W	8000h	23.5.16/ 906
788	INTC Priority Select Register (INTC_PSR916)	16	R/W	8000h	23.5.16/ 906
78A	INTC Priority Select Register (INTC_PSR917)	16	R/W	8000h	23.5.16/ 906
78C	INTC Priority Select Register (INTC_PSR918)	16	R/W	8000h	23.5.16/ 906
78E	INTC Priority Select Register (INTC_PSR919)	16	R/W	8000h	23.5.16/ 906
790	INTC Priority Select Register (INTC_PSR920)	16	R/W	8000h	23.5.16/ 906
792	INTC Priority Select Register (INTC_PSR921)	16	R/W	8000h	23.5.16/ 906
794	INTC Priority Select Register (INTC_PSR922)	16	R/W	8000h	23.5.16/ 906
796	INTC Priority Select Register (INTC_PSR923)	16	R/W	8000h	23.5.16/ 906
798	INTC Priority Select Register (INTC_PSR924)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
79A	INTC Priority Select Register (INTC_PSR925)	16	R/W	8000h	23.5.16/ 906
79C	INTC Priority Select Register (INTC_PSR926)	16	R/W	8000h	23.5.16/ 906
79E	INTC Priority Select Register (INTC_PSR927)	16	R/W	8000h	23.5.16/ 906
7A0	INTC Priority Select Register (INTC_PSR928)	16	R/W	8000h	23.5.16/ 906
7A2	INTC Priority Select Register (INTC_PSR929)	16	R/W	8000h	23.5.16/ 906
7A4	INTC Priority Select Register (INTC_PSR930)	16	R/W	8000h	23.5.16/ 906
7A6	INTC Priority Select Register (INTC_PSR931)	16	R/W	8000h	23.5.16/ 906
7A8	INTC Priority Select Register (INTC_PSR932)	16	R/W	8000h	23.5.16/ 906
7AA	INTC Priority Select Register (INTC_PSR933)	16	R/W	8000h	23.5.16/ 906
7AC	INTC Priority Select Register (INTC_PSR934)	16	R/W	8000h	23.5.16/ 906
7AE	INTC Priority Select Register (INTC_PSR935)	16	R/W	8000h	23.5.16/ 906
7B0	INTC Priority Select Register (INTC_PSR936)	16	R/W	8000h	23.5.16/ 906
7B2	INTC Priority Select Register (INTC_PSR937)	16	R/W	8000h	23.5.16/ 906
7B4	INTC Priority Select Register (INTC_PSR938)	16	R/W	8000h	23.5.16/ 906
7B6	INTC Priority Select Register (INTC_PSR939)	16	R/W	8000h	23.5.16/ 906
7B8	INTC Priority Select Register (INTC_PSR940)	16	R/W	8000h	23.5.16/ 906
7BA	INTC Priority Select Register (INTC_PSR941)	16	R/W	8000h	23.5.16/ 906
7BC	INTC Priority Select Register (INTC_PSR942)	16	R/W	8000h	23.5.16/ 906
7BE	INTC Priority Select Register (INTC_PSR943)	16	R/W	8000h	23.5.16/ 906
7C0	INTC Priority Select Register (INTC_PSR944)	16	R/W	8000h	23.5.16/ 906
7C2	INTC Priority Select Register (INTC_PSR945)	16	R/W	8000h	23.5.16/ 906
7C4	INTC Priority Select Register (INTC_PSR946)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7C6	INTC Priority Select Register (INTC_PSR947)	16	R/W	8000h	23.5.16/ 906
7C8	INTC Priority Select Register (INTC_PSR948)	16	R/W	8000h	23.5.16/ 906
7CA	INTC Priority Select Register (INTC_PSR949)	16	R/W	8000h	23.5.16/ 906
7CC	INTC Priority Select Register (INTC_PSR950)	16	R/W	8000h	23.5.16/ 906
7CE	INTC Priority Select Register (INTC_PSR951)	16	R/W	8000h	23.5.16/ 906
7D0	INTC Priority Select Register (INTC_PSR952)	16	R/W	8000h	23.5.16/ 906
7D2	INTC Priority Select Register (INTC_PSR953)	16	R/W	8000h	23.5.16/ 906
7D4	INTC Priority Select Register (INTC_PSR954)	16	R/W	8000h	23.5.16/ 906
7D6	INTC Priority Select Register (INTC_PSR955)	16	R/W	8000h	23.5.16/ 906
7D8	INTC Priority Select Register (INTC_PSR956)	16	R/W	8000h	23.5.16/ 906
7DA	INTC Priority Select Register (INTC_PSR957)	16	R/W	8000h	23.5.16/ 906
7DC	INTC Priority Select Register (INTC_PSR958)	16	R/W	8000h	23.5.16/ 906
7DE	INTC Priority Select Register (INTC_PSR959)	16	R/W	8000h	23.5.16/ 906
7E0	INTC Priority Select Register (INTC_PSR960)	16	R/W	8000h	23.5.16/ 906
7E2	INTC Priority Select Register (INTC_PSR961)	16	R/W	8000h	23.5.16/ 906
7E4	INTC Priority Select Register (INTC_PSR962)	16	R/W	8000h	23.5.16/ 906
7E6	INTC Priority Select Register (INTC_PSR963)	16	R/W	8000h	23.5.16/ 906
7E8	INTC Priority Select Register (INTC_PSR964)	16	R/W	8000h	23.5.16/ 906
7EA	INTC Priority Select Register (INTC_PSR965)	16	R/W	8000h	23.5.16/ 906
7EC	INTC Priority Select Register (INTC_PSR966)	16	R/W	8000h	23.5.16/ 906
7EE	INTC Priority Select Register (INTC_PSR967)	16	R/W	8000h	23.5.16/ 906
7F0	INTC Priority Select Register (INTC_PSR968)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7F2	INTC Priority Select Register (INTC_PSR969)	16	R/W	8000h	23.5.16/ 906
7F4	INTC Priority Select Register (INTC_PSR970)	16	R/W	8000h	23.5.16/ 906
7F6	INTC Priority Select Register (INTC_PSR971)	16	R/W	8000h	23.5.16/ 906
7F8	INTC Priority Select Register (INTC_PSR972)	16	R/W	8000h	23.5.16/ 906
7FA	INTC Priority Select Register (INTC_PSR973)	16	R/W	8000h	23.5.16/ 906
7FC	INTC Priority Select Register (INTC_PSR974)	16	R/W	8000h	23.5.16/ 906
7FE	INTC Priority Select Register (INTC_PSR975)	16	R/W	8000h	23.5.16/ 906
800	INTC Priority Select Register (INTC_PSR976)	16	R/W	8000h	23.5.16/ 906
802	INTC Priority Select Register (INTC_PSR977)	16	R/W	8000h	23.5.16/ 906
804	INTC Priority Select Register (INTC_PSR978)	16	R/W	8000h	23.5.16/ 906
806	INTC Priority Select Register (INTC_PSR979)	16	R/W	8000h	23.5.16/ 906
808	INTC Priority Select Register (INTC_PSR980)	16	R/W	8000h	23.5.16/ 906
80A	INTC Priority Select Register (INTC_PSR981)	16	R/W	8000h	23.5.16/ 906
80C	INTC Priority Select Register (INTC_PSR982)	16	R/W	8000h	23.5.16/ 906
80E	INTC Priority Select Register (INTC_PSR983)	16	R/W	8000h	23.5.16/ 906
810	INTC Priority Select Register (INTC_PSR984)	16	R/W	8000h	23.5.16/ 906
812	INTC Priority Select Register (INTC_PSR985)	16	R/W	8000h	23.5.16/ 906
814	INTC Priority Select Register (INTC_PSR986)	16	R/W	8000h	23.5.16/ 906
816	INTC Priority Select Register (INTC_PSR987)	16	R/W	8000h	23.5.16/ 906
818	INTC Priority Select Register (INTC_PSR988)	16	R/W	8000h	23.5.16/ 906
81A	INTC Priority Select Register (INTC_PSR989)	16	R/W	8000h	23.5.16/ 906
81C	INTC Priority Select Register (INTC_PSR990)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
81E	INTC Priority Select Register (INTC_PSR991)	16	R/W	8000h	23.5.16/ 906
820	INTC Priority Select Register (INTC_PSR992)	16	R/W	8000h	23.5.16/ 906
822	INTC Priority Select Register (INTC_PSR993)	16	R/W	8000h	23.5.16/ 906
824	INTC Priority Select Register (INTC_PSR994)	16	R/W	8000h	23.5.16/ 906
826	INTC Priority Select Register (INTC_PSR995)	16	R/W	8000h	23.5.16/ 906
828	INTC Priority Select Register (INTC_PSR996)	16	R/W	8000h	23.5.16/ 906
82A	INTC Priority Select Register (INTC_PSR997)	16	R/W	8000h	23.5.16/ 906
82C	INTC Priority Select Register (INTC_PSR998)	16	R/W	8000h	23.5.16/ 906
82E	INTC Priority Select Register (INTC_PSR999)	16	R/W	8000h	23.5.16/ 906
830	INTC Priority Select Register (INTC_PSR1000)	16	R/W	8000h	23.5.16/ 906
832	INTC Priority Select Register (INTC_PSR1001)	16	R/W	8000h	23.5.16/ 906
834	INTC Priority Select Register (INTC_PSR1002)	16	R/W	8000h	23.5.16/ 906
836	INTC Priority Select Register (INTC_PSR1003)	16	R/W	8000h	23.5.16/ 906
838	INTC Priority Select Register (INTC_PSR1004)	16	R/W	8000h	23.5.16/ 906
83A	INTC Priority Select Register (INTC_PSR1005)	16	R/W	8000h	23.5.16/ 906
83C	INTC Priority Select Register (INTC_PSR1006)	16	R/W	8000h	23.5.16/ 906
83E	INTC Priority Select Register (INTC_PSR1007)	16	R/W	8000h	23.5.16/ 906
840	INTC Priority Select Register (INTC_PSR1008)	16	R/W	8000h	23.5.16/ 906
842	INTC Priority Select Register (INTC_PSR1009)	16	R/W	8000h	23.5.16/ 906
844	INTC Priority Select Register (INTC_PSR1010)	16	R/W	8000h	23.5.16/ 906
846	INTC Priority Select Register (INTC_PSR1011)	16	R/W	8000h	23.5.16/ 906
848	INTC Priority Select Register (INTC_PSR1012)	16	R/W	8000h	23.5.16/ 906

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
84A	INTC Priority Select Register (INTC_PSR1013)	16	R/W	8000h	23.5.16/ 906
84C	INTC Priority Select Register (INTC_PSR1014)	16	R/W	8000h	23.5.16/ 906
84E	INTC Priority Select Register (INTC_PSR1015)	16	R/W	8000h	23.5.16/ 906
850	INTC Priority Select Register (INTC_PSR1016)	16	R/W	8000h	23.5.16/ 906
852	INTC Priority Select Register (INTC_PSR1017)	16	R/W	8000h	23.5.16/ 906
854	INTC Priority Select Register (INTC_PSR1018)	16	R/W	8000h	23.5.16/ 906
856	INTC Priority Select Register (INTC_PSR1019)	16	R/W	8000h	23.5.16/ 906
858	INTC Priority Select Register (INTC_PSR1020)	16	R/W	8000h	23.5.16/ 906
85A	INTC Priority Select Register (INTC_PSR1021)	16	R/W	8000h	23.5.16/ 906
85C	INTC Priority Select Register (INTC_PSR1022)	16	R/W	8000h	23.5.16/ 906
85E	INTC Priority Select Register (INTC_PSR1023)	16	R/W	8000h	23.5.16/ 906

23.5.1 INTC Block Configuration Register (INTC_BCR)

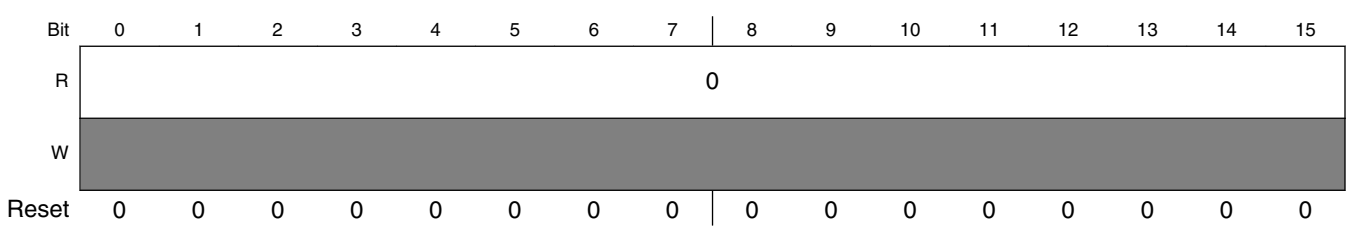
The Block Configuration Register is used to configure options of the INTC.

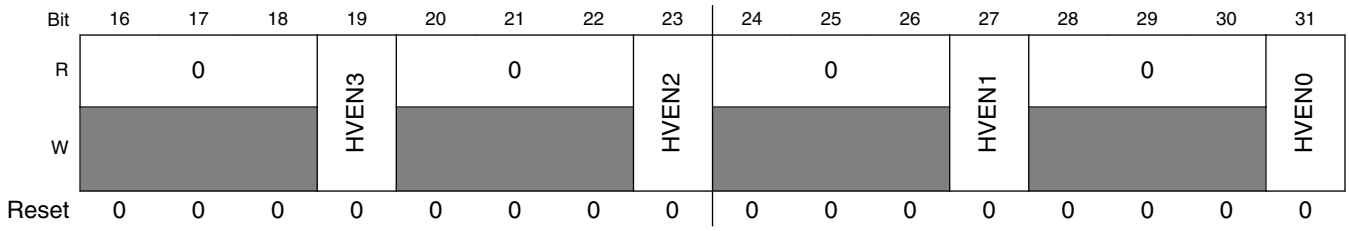
The master ID is used to determine if this register can be written by a given processor. Only the processor with master ID zero will be allowed write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] has write access, otherwise, a termination error is asserted.

Address: 0h base + 0h offset = 0h





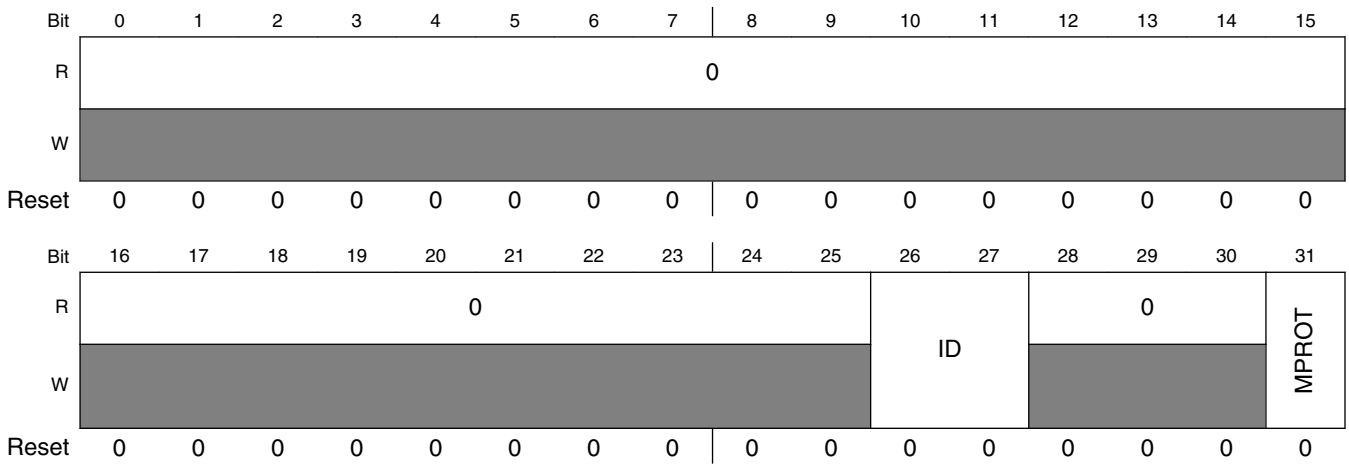
INTC_BCR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 HVEN3	Hardware vector enable for processor 3. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HVEN2	Hardware vector enable for processor 2. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 HVEN1	Hardware vector enable for processor 1. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVEN0	Hardware vector enable for processor 0. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

23.5.2 INTC Master Protection Register (INTC_MPROT)

The Master Protection Register is used to protect cores from writing to other cores INTC registers.

Address: 0h base + 4h offset = 4h



INTC_MPROT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 ID	ID This register can only be updated if MPROT=0 or if processor master ID=ID. When MPROT is set, the ID should also be updated to indicate which master can write to any register in the INTC memory map. Only the field values related to core bus masters can be selected. Non-core bus masters cannot be selected and would result in a bus access error. 00 master0 (reset default) 01 master1 10 master2 11 master3
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MPROT	Master Protection Controls what programming space a given master can write to. If MPROT is disabled, then all masters can write to any valid entry of the INTC memory map. If MPROT is enabled, INTC_MPROT[ID] can write to any valid entry of the INTC memory map but other masters can only write to their assigned resources. Once MPROT is enabled by a master, only that master may modify INTC_MPROT. 0 Master programming space protection disabled 1 Master programming space protection enabled

23.5.3 INTC Current Priority Register for Processor 0 (INTC_CPR0)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

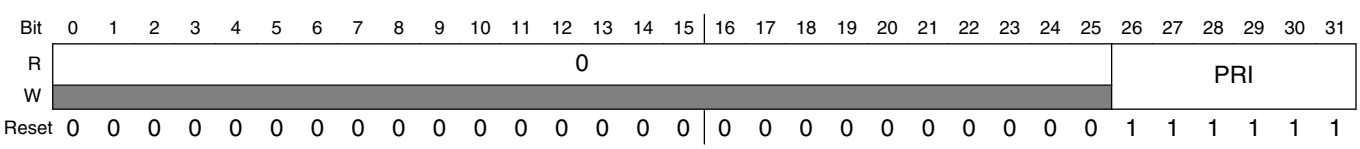
If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 10h offset = 10h



INTC_CPR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

23.5.4 INTC Current Priority Register for Processor 1 (INTC_CPR1)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										PRI					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

INTC_CPR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

23.5.5 INTC Current Priority Register for Processor 2 (INTC_CPR2)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

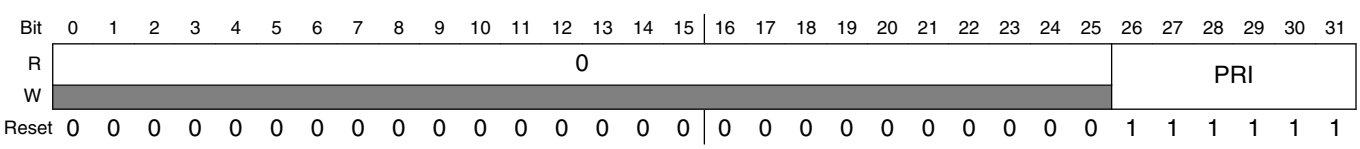
If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 18h offset = 18h



INTC_CPR2 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

23.5.6 INTC Current Priority Register for Processor 3 (INTC_CPR3)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

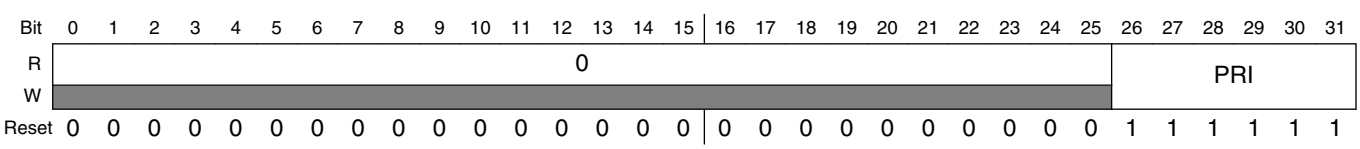
If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 1Ch offset = 1Ch



INTC_CPR3 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

23.5.7 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

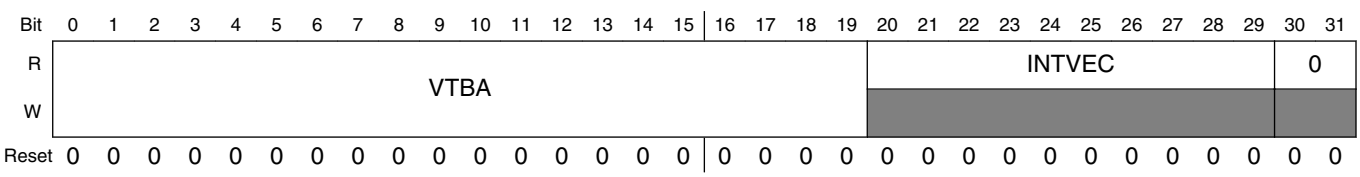
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 20h offset = 20h



INTC_IACKR0 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.5.8 INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

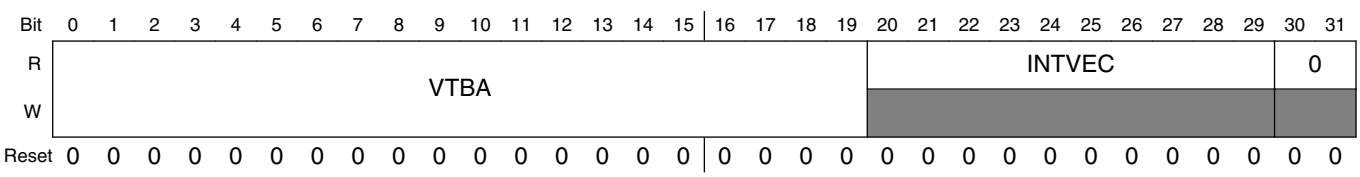
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 24h offset = 24h



INTC_IACKR1 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.5.9 INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

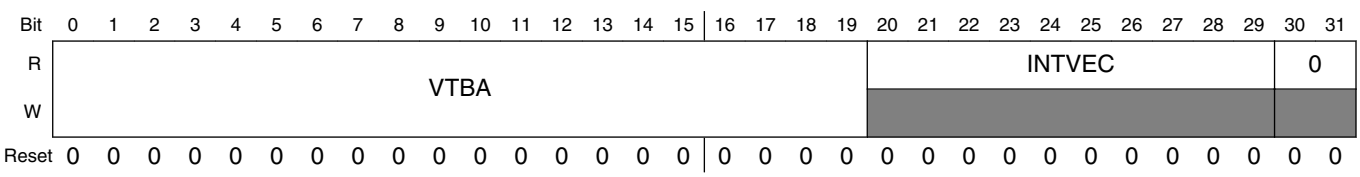
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 28h offset = 28h



INTC_IACKR2 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.5.10 INTC Interrupt Acknowledge Register for Processor 3 (INTC_IACKR3)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

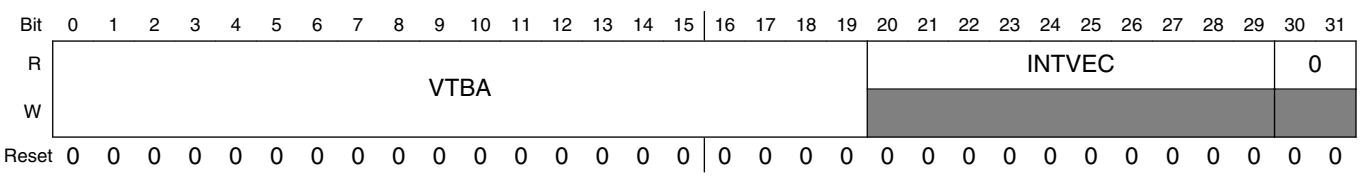
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 2Ch offset = 2Ch



INTC_IACKR3 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.5.11 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)

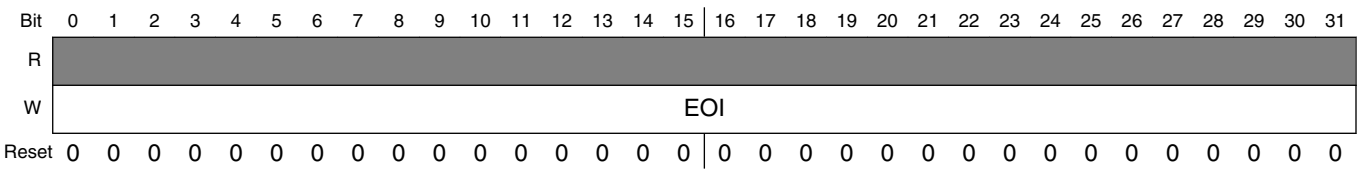
Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 30h offset = 30h



INTC_EOIR0 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

23.5.12 INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)

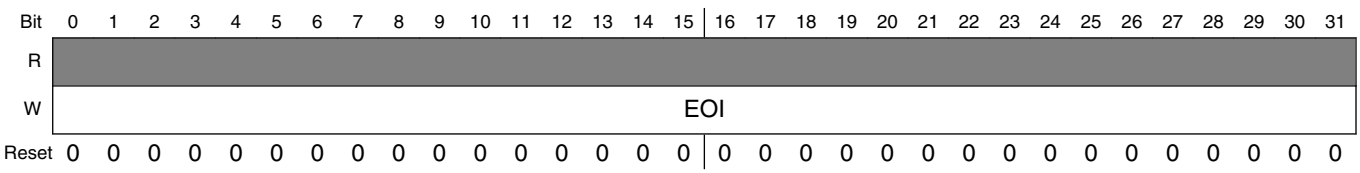
Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 34h offset = 34h



INTC_EOIR1 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

23.5.13 INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)

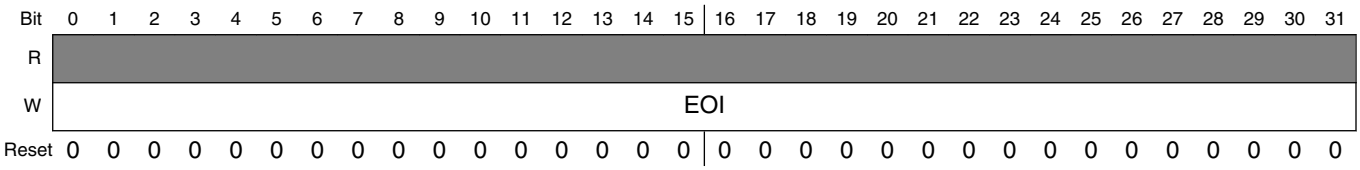
Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 38h offset = 38h



INTC_EOIR2 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

23.5.14 INTC End Of Interrupt Register for Processor 3 (INTC_EOIR3)

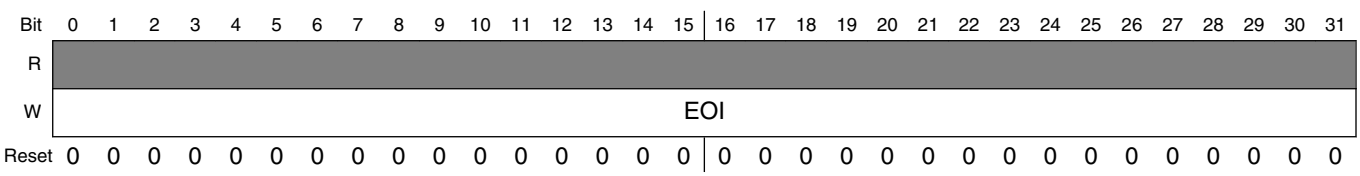
Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 3Ch offset = 3Ch



INTC_EOIR3 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

23.5.15 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIRn registers. The CLR bit will only be writable by the processor which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SEL]. An attempt by any other processor to write the CLR bit and not the SET bit will result in a termination error.

See Figure 23-2 for limitations on writing to this register.

NOTE

Since the four software setable interrupts in a word can be assigned to different processors, the cores with non-zero master IDs must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIRn registers.

If INTC_MPROT[MPROT]=0 (disabled), all processors can write to the CLR bit.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor can write to the CLR bit, otherwise, a termination error is asserted.

NOTE

If INTC_MPROT[MPROT] is enabled, processors with master ID != INTC_MPROT[ID] must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Address: 0h base + 40h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	0						0	CLR
Write							SET	w1c
Reset	0	0	0	0	0	0	0	0

INTC_SSCIR*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SET	Set flag bits. Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bits. CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

23.5.16 INTC Priority Select Register (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32-1023, and their priorities are configured in INTC_PSR32-1023, respectively.

Only the processor with master ID zero will be allowed to write the INTC_PSRn[PRC_SEL,SWT]. When writing the INTC_PSR[PRI], only the processor with master ID 0 and the processor who's master ID matches the setting of the INTC_PSRn[PRC_SEL] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the PSRn[PRIn] will result in a termination error.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access to INTC_PSRn[PRC_SEL,SWT], otherwise, a termination error is asserted. When writing the INTC_PSRn[PRI], only the processor with master ID (0-3) = INTC_MPROT[ID] and

the processor whose master ID (0-3) matches the setting of the INTC_PSRn[PRC_SEL] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the INTC_PSRn[PRI] will result in a termination error.

See Figure 23-2 for limitations on writing to this register.

NOTE

Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

NOTE

Interrupts are not detected while the PSRs are being written.

NOTE

If INTC_MPROT[MPROT] is enabled, the core with master ID (0-3) = INTC_MPROT[ID] will be able to write the INTC_PSRn with either byte, half-word or word operations. All other master IDs are restricted to byte operations to the INTC_PSRn[PRI] bits (i.e the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SEL] (i.e. the even byte) will result in a termination error.

NOTE

The core with master ID 0 will be able to write the INTC_PSRn with either byte, half-word or word operations. All other master IDs are restricted to byte operations to the PSRn[PRIn] bits (i.e the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SEL] (i.e. the even byte) will result in a termination error.

Address: 0h base + 60h offset + (2d × i), where i=0d to 1023d

Bit	0	1	2	3	4	5	6	7
Read	PRC_ SELN0	PRC_ SELN1	PRC_ SELN2	PRC_ SELN3	0			SWTN
Write								
Reset	1	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		PRIN					
Write								
Reset	0	0	0	0	0	0	0	0

INTC_PSR*n* field descriptions

Field	Description
0 PRC_SELN0	Processor select 0. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 0. 0 Interrupt request not sent to processor 0 1 Interrupt request sent to processor 0
1 PRC_SELN1	Processor select 1. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 1. 0 Interrupt request not sent to processor 1 1 Interrupt request sent to processor 1
2 PRC_SELN2	Processor select 2. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 2. 0 Interrupt request not sent to processor 2 1 Interrupt request sent to processor 2
3 PRC_SELN3	Processor select 3. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 3. 0 Interrupt request not sent to processor 3 1 Interrupt request sent to processor 3
4–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWTN	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 PRIN	Priority select. Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

23.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

23.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI*n* value in INTC_PSR*n* is higher than the PRI value in INTC_CPR*n*, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated INTC_CPR*n* is updated with the corresponding PRI*n* value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

23.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

23.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any Processor n will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the Processor n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SELn] bits.

23.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–31 are assigned vectors 0–31, respectively. The peripheral interrupt requests are assigned vectors from 32 to a number as high as needed to cover all peripheral interrupt requests.

23.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*n* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR*n*. The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

23.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in Figure 23-1 are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR*n*, and if in hardware vector mode, for the interrupt vector provided to the processor.

23.6.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

23.6.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

23.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

23.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR*n*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*n*, or the PRI value in the associated INTC_CPR*n* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR*n* when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in the associated INTC_CPR*n*.

Another function of the comparator is to signal an update of the INTC_IACKR*n* with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR*n* occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

23.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR*n*. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR*n* does not need to be loaded from the associated INTC_CPR*n* and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated INTC_CPR*n*.

The PRI value in the associated INTC_CPR*n* is pushed onto the LIFO when the associated INTC_IACKR*n* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*n* whenever the associated INTC_EOIR*n* is written. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

Although the INTC supports up to 64 priorities, an ISR executing with PRI in the INTC_CPR*n* equal to 63 will not be preempted. Therefore, the LIFO supports the stacking of 63 priorities. However, the LIFO is only 62 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 63 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

23.6.3 Handshaking with processor

This section describes handshaking with the processor.

23.6.3.1 Software vector mode handshaking

This section describes the software vector mode handshaking.

23.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to Figure 23-3.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 23-3. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in Software vector mode.

23.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*n* must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR*n*. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR*n* is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

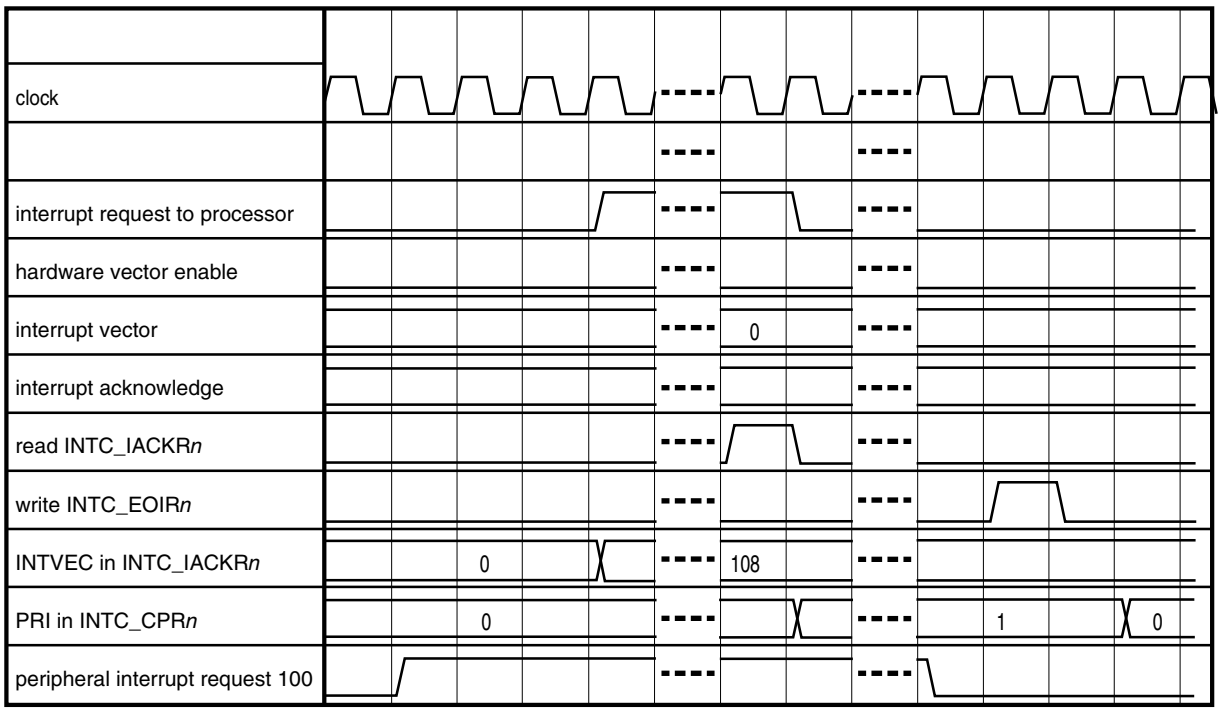


Figure 23-3. Timing diagram of software vector mode handshaking

23.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 23-4. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*n*. The rest of the handshaking is described in Hardware vector mode.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*n*, is the same as in software vector mode. See End of interrupt exception handler.

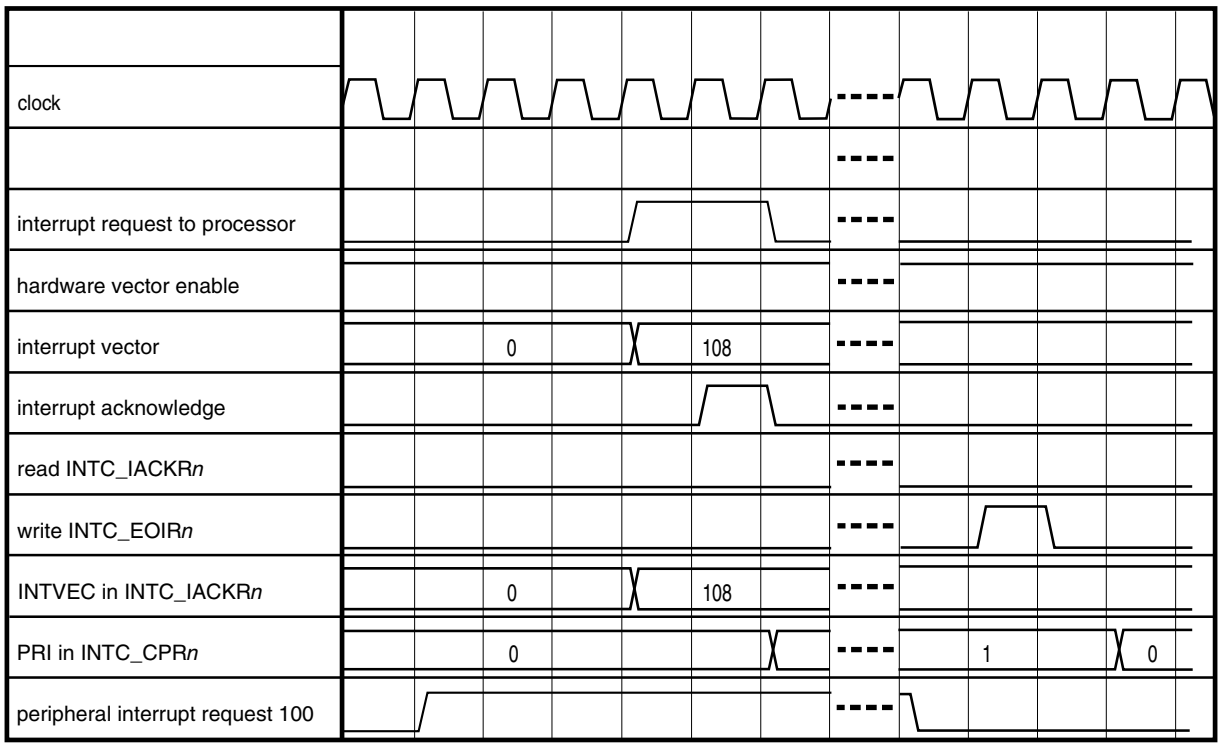


Figure 23-4. Timing diagram for hardware vector mode handshaking

23.7 Initialization/application information

This section contains initialization/application information.

23.7.1 Initialization flow

After exiting reset, all of the PRI*n* and PRC_SEL*n* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPRn registers is 63. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- interrupt_request_initialization:
  - Configure HVENn in INTC_BCR.
  - Configure VTBAn in INTC_IACKRn.
  - Raise the PRIn fields and set the PRC_SELn fields to the desired processor in INTC_PSRn.
  - Set the enable bits or clear the mask bits for the peripheral interrupt requests.

- Lower PRI in INTC_CPRn to zero.
- Enable processor(s) recognition of interrupts.

23.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

23.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b        interrupt_exception_handler_continued# 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1 (Power
Architecture Save/Restore registers) (not shown)

lis        r3,INTC_IACKRn@ha        # form adjusted upper half of INTC_IACKRn address
lwz        r3,INTC_IACKRn@l(r3)     # load INTC_IACKRn, which clears request to processor
lwz        r3,0x0(r3)                # load address of ISR from vector table
wrteei    1                        # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded Binary
Application Interface) (not shown)

mtlr    r3                         # move the INTC_IACKRn address into the link register
blrl                              # branch to ISR; link register updated with epilog
                                  # address
epilog:
code to restore most of context required by Power Architecture EABI (not shown)
# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                              # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                   # form 0 to write to INTC_EOIRn
wrteei    0                        # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)        # store to INTC_EOIRn, informing INTC to lower priority
code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
        .
        .
        .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023
ISRn:
code to service the interrupt event (not shown)
```

```
code to clear flag bit which drives interrupt request to INTC (not shown)
blr                                 # return to epilog
```

23.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b        interrupt_exception_handler_continuedn# 16 bytes available, branch to continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1 (not shown)

wrteei    1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl        ISRn                # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not shown)

# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                             # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                # form 0 to write to INTC_EOIRn
wrteei    0                    # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)    # store to INTC_EOIRn, informing INTC to lower priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr        # branch to epilog
```

23.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*n* having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR*n* priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR*n* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPRn priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR*n* while the shared resource is being accessed.

An ISR whose PRI*n* in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

23.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in the following table shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 23-1. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1

Table continues on the next page...

Table 23-1. Order of ISR execution example (continued)

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

23.7.5 Priority ceiling protocol

This section describes the priority ceiling protocol.

23.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR*n*) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR*n* to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*n* to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

23.7.5.2 Ensuring coherency

This section discusses how to ensure coherency.

23.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR*n*. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR*n* can be made by those system services with the code sequence:

```
GetResource:
        wrteei 0                # disable external interrupts to the Processor
        raise PRI                # Write to CPR, cache inhibit, guarded
        mbar                    # flush out writes from store buffer
        wrteei 1                # enable external interrupts to the Processor
        isync                    # re-fetch Processor pipeline

ReleaseResource:
        mbar                    # flush out writes from store buffer
        wrteei 0                # disable external interrupts to the Processor
        lower PRI                # Write to CPR, cache inhibit, guarded
        wrteei 1                # enable external interrupts to the Processor
```

23.7.5.2.1.1 Interrupt request to processor

Referencing Figure 23-1, the interrupt request logic to the processor is shown in the following figure.

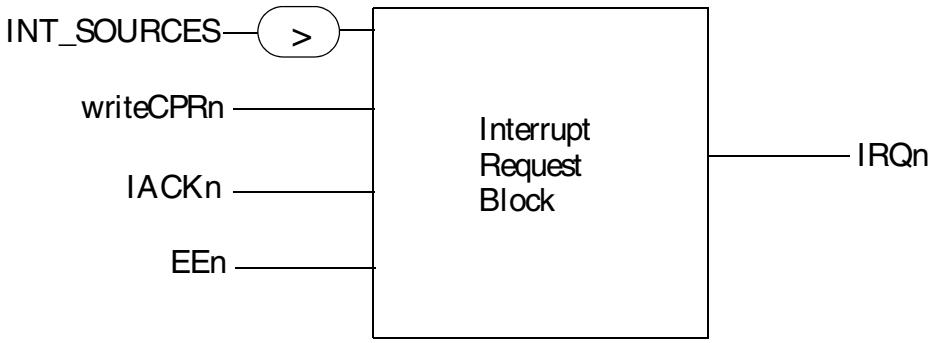


Figure 23-5. Interrupt Request Block Diagram

Assuming that there are no IRQn deasserted (no pending interrupt requests to processor n), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the INTC_CPRn, then interrupt request to the processor (IRQn) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACK) is asserted
- If the EEn bit has been cleared by the wrteei instruction (see the code sequence in Interrupt with blocked priority), IRQn will be re-evaluated when processor n writes to the INTC_CPRn while processor n ignores IRQn

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the INTC_CPRn.

23.7.5.2.2 Raised priority preserved

Before the instruction after the GetResource system service executes, all pending transactions have completed. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. The shared coherent data block now can be accessed coherently. The following figure shows the timing diagram for this scenario, and the following table explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

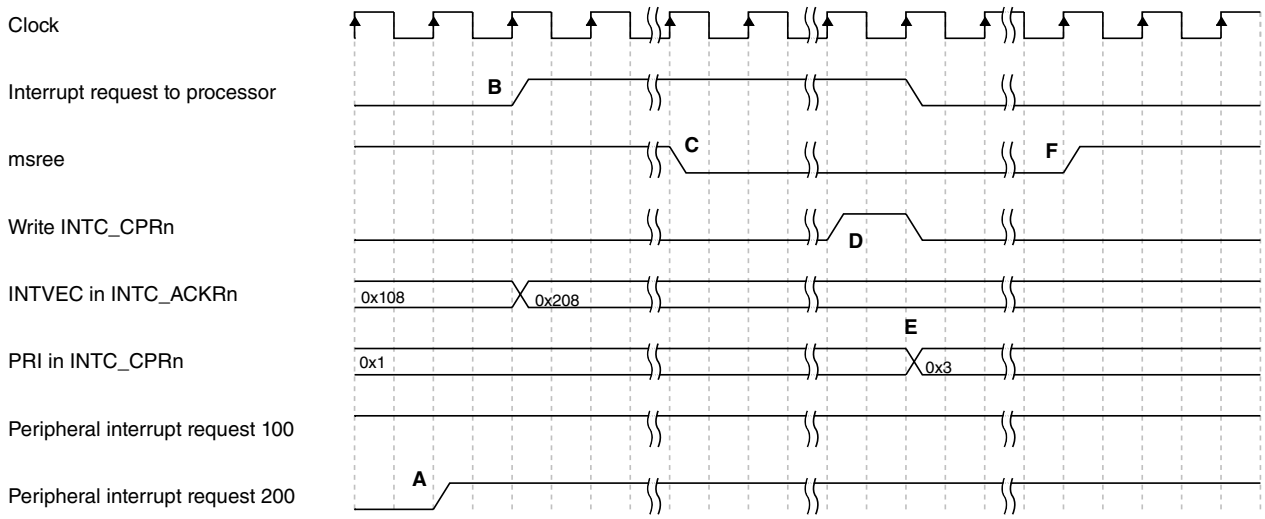


Figure 23-6. Timing diagram of raised priority preserved

Table 23-2. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 of priority 2 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	Write to msree bit to disable external interrupts
D	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
E	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
F	Write to msree bit to enable external interrupts

23.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

23.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

23.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRIn value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR*n* with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit

in INTC_SSCIR*n*. Writing a '1' to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI*n* value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

23.7.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

23.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in Scheduling a lower priority portion of an ISR) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note

Lowering the PRI value in INTC_CPR*n* within an ISR to below the ISR's corresponding PRI value in INTC_PSR*n* allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

23.7.9 Negating an interrupt request outside of its ISR

This section discusses the negation of an interrupt service request outside of its ISR.

23.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

23.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

23.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI*n* values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to End of interrupt exception handler for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*n* value in INTC_PSR*n*.

23.7.10 Examining LIFO contents

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR*n*). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPR*n*, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
      wrteei 0                # disable processor recognition of external interrupts
      oldCPRn = INTC_CPRn;    # save INTC_CPRn
      (branch to the pop_lifo)

pop_lifo:
      store INTC_EIORn        # pop INTC_CPR from LIFO, examine PRI, etc...
      examine INTC_CPRn[PRI], and store onto stack
      if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
      branch to push_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
      load stacked PRI value and store to INTC_CPRn
      load INTC_IACKn         # IACK; push INTC_CPRn into LIFO
      if stacked PRI values are not depleted, branch to push_lifo

      INTC_CPRn    = oldCPRn; # restore original INTC_CPRn and re-evaluate pending external
interrupts
      wrteei 1                # enable processor recognition of interrupts
```

NOTE

Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

23.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chip-specific INTC information.

Chapter 24
Enhanced Direct Memory Access (eDMA)

24.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors for each of the 64 channels.

The following figure illustrates the eDMA module.

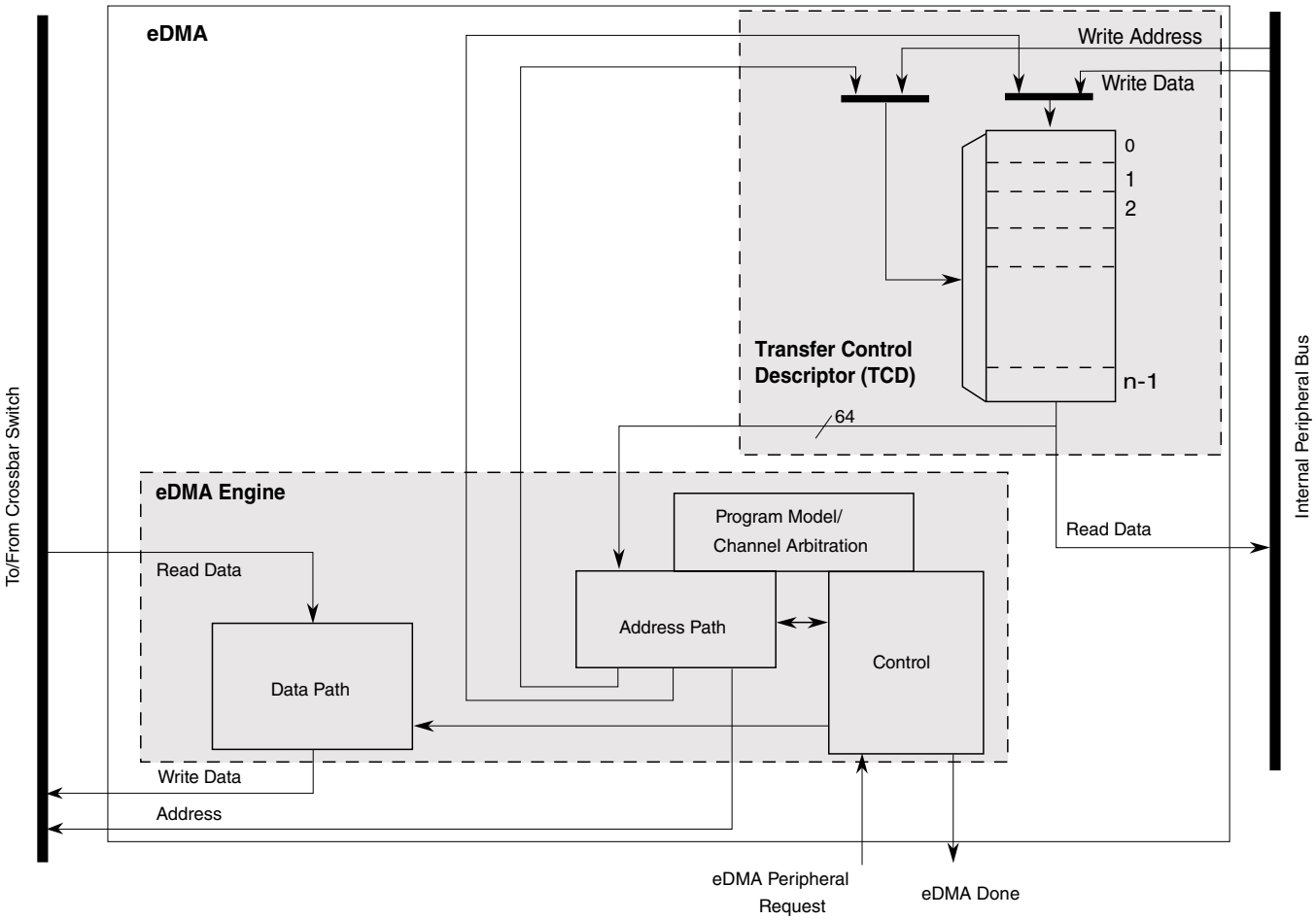


Figure 24-1. eDMA block diagram

24.1.1 Features

The eDMA is a highly-programmable data-transfer engine optimized to minimize the intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the data packet itself. The eDMA module features:

- All data movement via dual-address transfers—read from source, write to destination:

  Programmable source and destination addresses and transfer size, plus support for enhanced addressing modes

- 64-channel implementation that performs complex data transfers with minimal intervention from a host processor

    - Internal data buffer, used as temporary storage to support 16- and 32-byte burst transfers
    - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
    - 32-byte TCD stored in local memory for each channel
    - An inner data transfer loop defined by a minor byte transfer count
    - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
    - Explicit software initiation
    - Initiation via a channel-to-channel linking mechanism for continuous transfers
    - Peripheral-paced hardware requests (one per channel)
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
    - One interrupt per channel, optionally asserted at completion of major iteration count
    - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Optional support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software

Throughout this chapter, *n* is used to reference the channel number.

24.2 Modes of operation

The eDMA supports the following modes of operation.

24.2.1 Normal mode

In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.

24.2.2 Debug mode

DMA operation is configurable in Debug mode via the control register:

- If CR[EDBG] is cleared, the DMA continues to operate.
- If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

24.2.3 Wait mode

Before entering Wait mode the DMA attempts to complete its current transfer. After the transfer completes the device enters Wait mode.

24.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions.
- The second region corresponds to the local transfer control descriptor memory.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function.

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0 to channel 63. Each TCDn definition is presented as 11 registers of 16 or 32 bits.

Reading reserved bits in a register returns the value of zero and writes to reserved bits in a register are ignored. Reading or writing a reserved memory location generates a bus error.

eDMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (eDMA_CR)	32	R/W	0000_E400h	24.3.1/981
4	Error Status (eDMA_ES)	32	R	0000_0000h	24.3.2/983
8	Enable Request Register High (eDMA_ERQH)	32	R/W	0000_0000h	24.3.3/985
C	Enable Request Register Low (eDMA_ERQL)	32	R/W	0000_0000h	24.3.4/989
10	Enable Error Interrupt Register High (eDMA_EEIH)	32	R/W	0000_0000h	24.3.5/992
14	Enable Error Interrupt Register Low (eDMA_EEIL)	32	R/W	0000_0000h	24.3.6/996
18	Set Enable Request Register (eDMA_SERQ)	8	W	00h	24.3.7/999
19	Clear Enable Request Register (eDMA_CERQ)	8	W	00h	24.3.8/1000
1A	Set Enable Error Interrupt Register (eDMA_SEEI)	8	W	00h	24.3.9/1001
1B	Clear Enable Error Interrupt Register (eDMA_CEEI)	8	W	00h	24.3.10/ 1001
1C	Clear Interrupt Request Register (eDMA_CINT)	8	W	00h	24.3.11/ 1002
1D	Clear Error Register (eDMA_CERR)	8	W	00h	24.3.12/ 1003
1E	Set START Bit Register (eDMA_SSRT)	8	W	00h	24.3.13/ 1004
1F	Clear DONE Status Bit Register (eDMA_CDNE)	8	W	00h	24.3.14/ 1005
20	Interrupt Request Register High (eDMA_INTH)	32	w1c	0000_0000h	24.3.15/ 1005
24	Interrupt Request Register Low (eDMA_INTL)	32	w1c	0000_0000h	24.3.16/ 1009
28	Error Register High (eDMA_ERRH)	32	w1c	0000_0000h	24.3.17/ 1013
2C	Error Register Low (eDMA_ERRL)	32	w1c	0000_0000h	24.3.18/ 1017
30	Hardware Request Status Register High (eDMA_HRSH)	32	R/W	0000_0000h	24.3.19/ 1020
34	Hardware Request Status Register Low (eDMA_HRSL)	32	R/W	0000_0000h	24.3.20/ 1024
100	Channel Priority Register (eDMA_DCHPRI0)	8	R/W	See section	24.3.21/ 1027
101	Channel Priority Register (eDMA_DCHPRI1)	8	R/W	See section	24.3.21/ 1027
102	Channel Priority Register (eDMA_DCHPRI2)	8	R/W	See section	24.3.21/ 1027
103	Channel Priority Register (eDMA_DCHPRI3)	8	R/W	See section	24.3.21/ 1027

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
104	Channel Priority Register (eDMA_DCHPRI4)	8	R/W	See section	24.3.21/ 1027
105	Channel Priority Register (eDMA_DCHPRI5)	8	R/W	See section	24.3.21/ 1027
106	Channel Priority Register (eDMA_DCHPRI6)	8	R/W	See section	24.3.21/ 1027
107	Channel Priority Register (eDMA_DCHPRI7)	8	R/W	See section	24.3.21/ 1027
108	Channel Priority Register (eDMA_DCHPRI8)	8	R/W	See section	24.3.21/ 1027
109	Channel Priority Register (eDMA_DCHPRI9)	8	R/W	See section	24.3.21/ 1027
10A	Channel Priority Register (eDMA_DCHPRI10)	8	R/W	See section	24.3.21/ 1027
10B	Channel Priority Register (eDMA_DCHPRI11)	8	R/W	See section	24.3.21/ 1027
10C	Channel Priority Register (eDMA_DCHPRI12)	8	R/W	See section	24.3.21/ 1027
10D	Channel Priority Register (eDMA_DCHPRI13)	8	R/W	See section	24.3.21/ 1027
10E	Channel Priority Register (eDMA_DCHPRI14)	8	R/W	See section	24.3.21/ 1027
10F	Channel Priority Register (eDMA_DCHPRI15)	8	R/W	See section	24.3.21/ 1027
110	Channel Priority Register (eDMA_DCHPRI16)	8	R/W	See section	24.3.21/ 1027
111	Channel Priority Register (eDMA_DCHPRI17)	8	R/W	See section	24.3.21/ 1027
112	Channel Priority Register (eDMA_DCHPRI18)	8	R/W	See section	24.3.21/ 1027
113	Channel Priority Register (eDMA_DCHPRI19)	8	R/W	See section	24.3.21/ 1027
114	Channel Priority Register (eDMA_DCHPRI20)	8	R/W	See section	24.3.21/ 1027
115	Channel Priority Register (eDMA_DCHPRI21)	8	R/W	See section	24.3.21/ 1027
116	Channel Priority Register (eDMA_DCHPRI22)	8	R/W	See section	24.3.21/ 1027
117	Channel Priority Register (eDMA_DCHPRI23)	8	R/W	See section	24.3.21/ 1027
118	Channel Priority Register (eDMA_DCHPRI24)	8	R/W	See section	24.3.21/ 1027
119	Channel Priority Register (eDMA_DCHPRI25)	8	R/W	See section	24.3.21/ 1027

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11A	Channel Priority Register (eDMA_DCHPRI26)	8	R/W	See section	24.3.21/ 1027
11B	Channel Priority Register (eDMA_DCHPRI27)	8	R/W	See section	24.3.21/ 1027
11C	Channel Priority Register (eDMA_DCHPRI28)	8	R/W	See section	24.3.21/ 1027
11D	Channel Priority Register (eDMA_DCHPRI29)	8	R/W	See section	24.3.21/ 1027
11E	Channel Priority Register (eDMA_DCHPRI30)	8	R/W	See section	24.3.21/ 1027
11F	Channel Priority Register (eDMA_DCHPRI31)	8	R/W	See section	24.3.21/ 1027
120	Channel Priority Register (eDMA_DCHPRI32)	8	R/W	See section	24.3.21/ 1027
121	Channel Priority Register (eDMA_DCHPRI33)	8	R/W	See section	24.3.21/ 1027
122	Channel Priority Register (eDMA_DCHPRI34)	8	R/W	See section	24.3.21/ 1027
123	Channel Priority Register (eDMA_DCHPRI35)	8	R/W	See section	24.3.21/ 1027
124	Channel Priority Register (eDMA_DCHPRI36)	8	R/W	See section	24.3.21/ 1027
125	Channel Priority Register (eDMA_DCHPRI37)	8	R/W	See section	24.3.21/ 1027
126	Channel Priority Register (eDMA_DCHPRI38)	8	R/W	See section	24.3.21/ 1027
127	Channel Priority Register (eDMA_DCHPRI39)	8	R/W	See section	24.3.21/ 1027
128	Channel Priority Register (eDMA_DCHPRI40)	8	R/W	See section	24.3.21/ 1027
129	Channel Priority Register (eDMA_DCHPRI41)	8	R/W	See section	24.3.21/ 1027
12A	Channel Priority Register (eDMA_DCHPRI42)	8	R/W	See section	24.3.21/ 1027
12B	Channel Priority Register (eDMA_DCHPRI43)	8	R/W	See section	24.3.21/ 1027
12C	Channel Priority Register (eDMA_DCHPRI44)	8	R/W	See section	24.3.21/ 1027
12D	Channel Priority Register (eDMA_DCHPRI45)	8	R/W	See section	24.3.21/ 1027
12E	Channel Priority Register (eDMA_DCHPRI46)	8	R/W	See section	24.3.21/ 1027
12F	Channel Priority Register (eDMA_DCHPRI47)	8	R/W	See section	24.3.21/ 1027

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
130	Channel Priority Register (eDMA_DCHPRI48)	8	R/W	See section	24.3.21/ 1027
131	Channel Priority Register (eDMA_DCHPRI49)	8	R/W	See section	24.3.21/ 1027
132	Channel Priority Register (eDMA_DCHPRI50)	8	R/W	See section	24.3.21/ 1027
133	Channel Priority Register (eDMA_DCHPRI51)	8	R/W	See section	24.3.21/ 1027
134	Channel Priority Register (eDMA_DCHPRI52)	8	R/W	See section	24.3.21/ 1027
135	Channel Priority Register (eDMA_DCHPRI53)	8	R/W	See section	24.3.21/ 1027
136	Channel Priority Register (eDMA_DCHPRI54)	8	R/W	See section	24.3.21/ 1027
137	Channel Priority Register (eDMA_DCHPRI55)	8	R/W	See section	24.3.21/ 1027
138	Channel Priority Register (eDMA_DCHPRI56)	8	R/W	See section	24.3.21/ 1027
139	Channel Priority Register (eDMA_DCHPRI57)	8	R/W	See section	24.3.21/ 1027
13A	Channel Priority Register (eDMA_DCHPRI58)	8	R/W	See section	24.3.21/ 1027
13B	Channel Priority Register (eDMA_DCHPRI59)	8	R/W	See section	24.3.21/ 1027
13C	Channel Priority Register (eDMA_DCHPRI60)	8	R/W	See section	24.3.21/ 1027
13D	Channel Priority Register (eDMA_DCHPRI61)	8	R/W	See section	24.3.21/ 1027
13E	Channel Priority Register (eDMA_DCHPRI62)	8	R/W	See section	24.3.21/ 1027
13F	Channel Priority Register (eDMA_DCHPRI63)	8	R/W	See section	24.3.21/ 1027
140	Channel Master ID Register (eDMA_DCHMID0)	8	R/W	00h	24.3.22/ 1029
141	Channel Master ID Register (eDMA_DCHMID1)	8	R/W	00h	24.3.22/ 1029
142	Channel Master ID Register (eDMA_DCHMID2)	8	R/W	00h	24.3.22/ 1029
143	Channel Master ID Register (eDMA_DCHMID3)	8	R/W	00h	24.3.22/ 1029
144	Channel Master ID Register (eDMA_DCHMID4)	8	R/W	00h	24.3.22/ 1029
145	Channel Master ID Register (eDMA_DCHMID5)	8	R/W	00h	24.3.22/ 1029

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
146	Channel Master ID Register (eDMA_DCHMID6)	8	R/W	00h	24.3.22/ 1029
147	Channel Master ID Register (eDMA_DCHMID7)	8	R/W	00h	24.3.22/ 1029
148	Channel Master ID Register (eDMA_DCHMID8)	8	R/W	00h	24.3.22/ 1029
149	Channel Master ID Register (eDMA_DCHMID9)	8	R/W	00h	24.3.22/ 1029
14A	Channel Master ID Register (eDMA_DCHMID10)	8	R/W	00h	24.3.22/ 1029
14B	Channel Master ID Register (eDMA_DCHMID11)	8	R/W	00h	24.3.22/ 1029
14C	Channel Master ID Register (eDMA_DCHMID12)	8	R/W	00h	24.3.22/ 1029
14D	Channel Master ID Register (eDMA_DCHMID13)	8	R/W	00h	24.3.22/ 1029
14E	Channel Master ID Register (eDMA_DCHMID14)	8	R/W	00h	24.3.22/ 1029
14F	Channel Master ID Register (eDMA_DCHMID15)	8	R/W	00h	24.3.22/ 1029
150	Channel Master ID Register (eDMA_DCHMID16)	8	R/W	00h	24.3.22/ 1029
151	Channel Master ID Register (eDMA_DCHMID17)	8	R/W	00h	24.3.22/ 1029
152	Channel Master ID Register (eDMA_DCHMID18)	8	R/W	00h	24.3.22/ 1029
153	Channel Master ID Register (eDMA_DCHMID19)	8	R/W	00h	24.3.22/ 1029
154	Channel Master ID Register (eDMA_DCHMID20)	8	R/W	00h	24.3.22/ 1029
155	Channel Master ID Register (eDMA_DCHMID21)	8	R/W	00h	24.3.22/ 1029
156	Channel Master ID Register (eDMA_DCHMID22)	8	R/W	00h	24.3.22/ 1029
157	Channel Master ID Register (eDMA_DCHMID23)	8	R/W	00h	24.3.22/ 1029
158	Channel Master ID Register (eDMA_DCHMID24)	8	R/W	00h	24.3.22/ 1029
159	Channel Master ID Register (eDMA_DCHMID25)	8	R/W	00h	24.3.22/ 1029
15A	Channel Master ID Register (eDMA_DCHMID26)	8	R/W	00h	24.3.22/ 1029
15B	Channel Master ID Register (eDMA_DCHMID27)	8	R/W	00h	24.3.22/ 1029

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15C	Channel Master ID Register (eDMA_DCHMID28)	8	R/W	00h	24.3.22/ 1029
15D	Channel Master ID Register (eDMA_DCHMID29)	8	R/W	00h	24.3.22/ 1029
15E	Channel Master ID Register (eDMA_DCHMID30)	8	R/W	00h	24.3.22/ 1029
15F	Channel Master ID Register (eDMA_DCHMID31)	8	R/W	00h	24.3.22/ 1029
160	Channel Master ID Register (eDMA_DCHMID32)	8	R/W	00h	24.3.22/ 1029
161	Channel Master ID Register (eDMA_DCHMID33)	8	R/W	00h	24.3.22/ 1029
162	Channel Master ID Register (eDMA_DCHMID34)	8	R/W	00h	24.3.22/ 1029
163	Channel Master ID Register (eDMA_DCHMID35)	8	R/W	00h	24.3.22/ 1029
164	Channel Master ID Register (eDMA_DCHMID36)	8	R/W	00h	24.3.22/ 1029
165	Channel Master ID Register (eDMA_DCHMID37)	8	R/W	00h	24.3.22/ 1029
166	Channel Master ID Register (eDMA_DCHMID38)	8	R/W	00h	24.3.22/ 1029
167	Channel Master ID Register (eDMA_DCHMID39)	8	R/W	00h	24.3.22/ 1029
168	Channel Master ID Register (eDMA_DCHMID40)	8	R/W	00h	24.3.22/ 1029
169	Channel Master ID Register (eDMA_DCHMID41)	8	R/W	00h	24.3.22/ 1029
16A	Channel Master ID Register (eDMA_DCHMID42)	8	R/W	00h	24.3.22/ 1029
16B	Channel Master ID Register (eDMA_DCHMID43)	8	R/W	00h	24.3.22/ 1029
16C	Channel Master ID Register (eDMA_DCHMID44)	8	R/W	00h	24.3.22/ 1029
16D	Channel Master ID Register (eDMA_DCHMID45)	8	R/W	00h	24.3.22/ 1029
16E	Channel Master ID Register (eDMA_DCHMID46)	8	R/W	00h	24.3.22/ 1029
16F	Channel Master ID Register (eDMA_DCHMID47)	8	R/W	00h	24.3.22/ 1029
170	Channel Master ID Register (eDMA_DCHMID48)	8	R/W	00h	24.3.22/ 1029
171	Channel Master ID Register (eDMA_DCHMID49)	8	R/W	00h	24.3.22/ 1029

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
172	Channel Master ID Register (eDMA_DCHMID50)	8	R/W	00h	24.3.22/ 1029
173	Channel Master ID Register (eDMA_DCHMID51)	8	R/W	00h	24.3.22/ 1029
174	Channel Master ID Register (eDMA_DCHMID52)	8	R/W	00h	24.3.22/ 1029
175	Channel Master ID Register (eDMA_DCHMID53)	8	R/W	00h	24.3.22/ 1029
176	Channel Master ID Register (eDMA_DCHMID54)	8	R/W	00h	24.3.22/ 1029
177	Channel Master ID Register (eDMA_DCHMID55)	8	R/W	00h	24.3.22/ 1029
178	Channel Master ID Register (eDMA_DCHMID56)	8	R/W	00h	24.3.22/ 1029
179	Channel Master ID Register (eDMA_DCHMID57)	8	R/W	00h	24.3.22/ 1029
17A	Channel Master ID Register (eDMA_DCHMID58)	8	R/W	00h	24.3.22/ 1029
17B	Channel Master ID Register (eDMA_DCHMID59)	8	R/W	00h	24.3.22/ 1029
17C	Channel Master ID Register (eDMA_DCHMID60)	8	R/W	00h	24.3.22/ 1029
17D	Channel Master ID Register (eDMA_DCHMID61)	8	R/W	00h	24.3.22/ 1029
17E	Channel Master ID Register (eDMA_DCHMID62)	8	R/W	00h	24.3.22/ 1029
17F	Channel Master ID Register (eDMA_DCHMID63)	8	R/W	00h	24.3.22/ 1029
1000	TCD Source Address (eDMA_TCD0_SADDR)	32	R/W	See section	24.3.23/ 1030
1004	TCD Transfer Attributes (eDMA_TCD0_ATTR)	16	R/W	See section	24.3.24/ 1030
1006	TCD Signed Source Address Offset (eDMA_TCD0_SOFF)	16	R/W	See section	24.3.25/ 1031
1008	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD0_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1008	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD0_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1008	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD0_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
100C	TCD Last Source Address Adjustment (eDMA_TCD0_SLAST)	32	R/W	See section	24.3.29/ 1035
1010	TCD Destination Address (eDMA_TCD0_DADDR)	32	R/W	See section	24.3.30/ 1035

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1014	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD0_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1014	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD0_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1016	TCD Signed Destination Address Offset (eDMA_TCD0_DOFF)	16	R/W	See section	24.3.33/ 1038
1018	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD0_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
101C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD0_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
101C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD0_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
101E	TCD Control and Status (eDMA_TCD0_CSR)	16	R/W	See section	24.3.37/ 1041
1020	TCD Source Address (eDMA_TCD1_SADDR)	32	R/W	See section	24.3.23/ 1030
1024	TCD Transfer Attributes (eDMA_TCD1_ATTR)	16	R/W	See section	24.3.24/ 1030
1026	TCD Signed Source Address Offset (eDMA_TCD1_SOFF)	16	R/W	See section	24.3.25/ 1031
1028	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD1_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1028	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD1_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1028	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD1_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
102C	TCD Last Source Address Adjustment (eDMA_TCD1_SLAST)	32	R/W	See section	24.3.29/ 1035
1030	TCD Destination Address (eDMA_TCD1_DADDR)	32	R/W	See section	24.3.30/ 1035
1034	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD1_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1034	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD1_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1036	TCD Signed Destination Address Offset (eDMA_TCD1_DOFF)	16	R/W	See section	24.3.33/ 1038
1038	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD1_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
103C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD1_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
103C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD1_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
103E	TCD Control and Status (eDMA_TCD1_CSR)	16	R/W	See section	24.3.37/ 1041

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1040	TCD Source Address (eDMA_TCD2_SADDR)	32	R/W	See section	24.3.23/ 1030
1044	TCD Transfer Attributes (eDMA_TCD2_ATTR)	16	R/W	See section	24.3.24/ 1030
1046	TCD Signed Source Address Offset (eDMA_TCD2_SOFF)	16	R/W	See section	24.3.25/ 1031
1048	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD2_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1048	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD2_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1048	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD2_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
104C	TCD Last Source Address Adjustment (eDMA_TCD2_SLAST)	32	R/W	See section	24.3.29/ 1035
1050	TCD Destination Address (eDMA_TCD2_DADDR)	32	R/W	See section	24.3.30/ 1035
1054	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD2_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1054	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD2_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1056	TCD Signed Destination Address Offset (eDMA_TCD2_DOFF)	16	R/W	See section	24.3.33/ 1038
1058	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD2_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
105C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD2_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
105C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD2_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
105E	TCD Control and Status (eDMA_TCD2_CSR)	16	R/W	See section	24.3.37/ 1041
1060	TCD Source Address (eDMA_TCD3_SADDR)	32	R/W	See section	24.3.23/ 1030
1064	TCD Transfer Attributes (eDMA_TCD3_ATTR)	16	R/W	See section	24.3.24/ 1030
1066	TCD Signed Source Address Offset (eDMA_TCD3_SOFF)	16	R/W	See section	24.3.25/ 1031
1068	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD3_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1068	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD3_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1068	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD3_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
106C	TCD Last Source Address Adjustment (eDMA_TCD3_SLAST)	32	R/W	See section	24.3.29/ 1035

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1070	TCD Destination Address (eDMA_TCD3_DADDR)	32	R/W	See section	24.3.30/ 1035
1074	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD3_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1074	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD3_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1076	TCD Signed Destination Address Offset (eDMA_TCD3_DOFF)	16	R/W	See section	24.3.33/ 1038
1078	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD3_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
107C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD3_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
107C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD3_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
107E	TCD Control and Status (eDMA_TCD3_CSR)	16	R/W	See section	24.3.37/ 1041
1080	TCD Source Address (eDMA_TCD4_SADDR)	32	R/W	See section	24.3.23/ 1030
1084	TCD Transfer Attributes (eDMA_TCD4_ATTR)	16	R/W	See section	24.3.24/ 1030
1086	TCD Signed Source Address Offset (eDMA_TCD4_SOFF)	16	R/W	See section	24.3.25/ 1031
1088	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD4_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1088	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD4_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1088	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD4_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
108C	TCD Last Source Address Adjustment (eDMA_TCD4_SLAST)	32	R/W	See section	24.3.29/ 1035
1090	TCD Destination Address (eDMA_TCD4_DADDR)	32	R/W	See section	24.3.30/ 1035
1094	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD4_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1094	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD4_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1096	TCD Signed Destination Address Offset (eDMA_TCD4_DOFF)	16	R/W	See section	24.3.33/ 1038
1098	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD4_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
109C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD4_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
109C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD4_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
109E	TCD Control and Status (eDMA_TCD4_CSR)	16	R/W	See section	24.3.37/ 1041
10A0	TCD Source Address (eDMA_TCD5_SADDR)	32	R/W	See section	24.3.23/ 1030
10A4	TCD Transfer Attributes (eDMA_TCD5_ATTR)	16	R/W	See section	24.3.24/ 1030
10A6	TCD Signed Source Address Offset (eDMA_TCD5_SOFF)	16	R/W	See section	24.3.25/ 1031
10A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD5_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
10A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD5_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
10A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD5_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
10AC	TCD Last Source Address Adjustment (eDMA_TCD5_SLAST)	32	R/W	See section	24.3.29/ 1035
10B0	TCD Destination Address (eDMA_TCD5_DADDR)	32	R/W	See section	24.3.30/ 1035
10B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD5_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
10B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD5_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
10B6	TCD Signed Destination Address Offset (eDMA_TCD5_DOFF)	16	R/W	See section	24.3.33/ 1038
10B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD5_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
10BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD5_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
10BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD5_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
10BE	TCD Control and Status (eDMA_TCD5_CSR)	16	R/W	See section	24.3.37/ 1041
10C0	TCD Source Address (eDMA_TCD6_SADDR)	32	R/W	See section	24.3.23/ 1030
10C4	TCD Transfer Attributes (eDMA_TCD6_ATTR)	16	R/W	See section	24.3.24/ 1030
10C6	TCD Signed Source Address Offset (eDMA_TCD6_SOFF)	16	R/W	See section	24.3.25/ 1031
10C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD6_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
10C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD6_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
10C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD6_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10CC	TCD Last Source Address Adjustment (eDMA_TCD6_SLAST)	32	R/W	See section	24.3.29/ 1035
10D0	TCD Destination Address (eDMA_TCD6_DADDR)	32	R/W	See section	24.3.30/ 1035
10D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD6_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
10D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD6_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
10D6	TCD Signed Destination Address Offset (eDMA_TCD6_DOFF)	16	R/W	See section	24.3.33/ 1038
10D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD6_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
10DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD6_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
10DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD6_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
10DE	TCD Control and Status (eDMA_TCD6_CSR)	16	R/W	See section	24.3.37/ 1041
10E0	TCD Source Address (eDMA_TCD7_SADDR)	32	R/W	See section	24.3.23/ 1030
10E4	TCD Transfer Attributes (eDMA_TCD7_ATTR)	16	R/W	See section	24.3.24/ 1030
10E6	TCD Signed Source Address Offset (eDMA_TCD7_SOFF)	16	R/W	See section	24.3.25/ 1031
10E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD7_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
10E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD7_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
10E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD7_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
10EC	TCD Last Source Address Adjustment (eDMA_TCD7_SLAST)	32	R/W	See section	24.3.29/ 1035
10F0	TCD Destination Address (eDMA_TCD7_DADDR)	32	R/W	See section	24.3.30/ 1035
10F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD7_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
10F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD7_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
10F6	TCD Signed Destination Address Offset (eDMA_TCD7_DOFF)	16	R/W	See section	24.3.33/ 1038
10F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD7_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
10FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD7_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD7_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
10FE	TCD Control and Status (eDMA_TCD7_CSR)	16	R/W	See section	24.3.37/ 1041
1100	TCD Source Address (eDMA_TCD8_SADDR)	32	R/W	See section	24.3.23/ 1030
1104	TCD Transfer Attributes (eDMA_TCD8_ATTR)	16	R/W	See section	24.3.24/ 1030
1106	TCD Signed Source Address Offset (eDMA_TCD8_SOFF)	16	R/W	See section	24.3.25/ 1031
1108	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD8_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1108	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD8_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1108	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD8_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
110C	TCD Last Source Address Adjustment (eDMA_TCD8_SLAST)	32	R/W	See section	24.3.29/ 1035
1110	TCD Destination Address (eDMA_TCD8_DADDR)	32	R/W	See section	24.3.30/ 1035
1114	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD8_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1114	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD8_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1116	TCD Signed Destination Address Offset (eDMA_TCD8_DOFF)	16	R/W	See section	24.3.33/ 1038
1118	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD8_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
111C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD8_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
111C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD8_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
111E	TCD Control and Status (eDMA_TCD8_CSR)	16	R/W	See section	24.3.37/ 1041
1120	TCD Source Address (eDMA_TCD9_SADDR)	32	R/W	See section	24.3.23/ 1030
1124	TCD Transfer Attributes (eDMA_TCD9_ATTR)	16	R/W	See section	24.3.24/ 1030
1126	TCD Signed Source Address Offset (eDMA_TCD9_SOFF)	16	R/W	See section	24.3.25/ 1031
1128	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD9_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1128	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD9_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1128	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD9_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
112C	TCD Last Source Address Adjustment (eDMA_TCD9_SLAST)	32	R/W	See section	24.3.29/ 1035
1130	TCD Destination Address (eDMA_TCD9_DADDR)	32	R/W	See section	24.3.30/ 1035
1134	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD9_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1134	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD9_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1136	TCD Signed Destination Address Offset (eDMA_TCD9_DOFF)	16	R/W	See section	24.3.33/ 1038
1138	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD9_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
113C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD9_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
113C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD9_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
113E	TCD Control and Status (eDMA_TCD9_CSR)	16	R/W	See section	24.3.37/ 1041
1140	TCD Source Address (eDMA_TCD10_SADDR)	32	R/W	See section	24.3.23/ 1030
1144	TCD Transfer Attributes (eDMA_TCD10_ATTR)	16	R/W	See section	24.3.24/ 1030
1146	TCD Signed Source Address Offset (eDMA_TCD10_SOFF)	16	R/W	See section	24.3.25/ 1031
1148	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD10_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1148	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD10_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1148	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD10_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
114C	TCD Last Source Address Adjustment (eDMA_TCD10_SLAST)	32	R/W	See section	24.3.29/ 1035
1150	TCD Destination Address (eDMA_TCD10_DADDR)	32	R/W	See section	24.3.30/ 1035
1154	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD10_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1154	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD10_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1156	TCD Signed Destination Address Offset (eDMA_TCD10_DOFF)	16	R/W	See section	24.3.33/ 1038
1158	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD10_DLASTSGA)	32	R/W	See section	24.3.34/ 1038

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
115C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD10_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
115C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD10_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
115E	TCD Control and Status (eDMA_TCD10_CSR)	16	R/W	See section	24.3.37/ 1041
1160	TCD Source Address (eDMA_TCD11_SADDR)	32	R/W	See section	24.3.23/ 1030
1164	TCD Transfer Attributes (eDMA_TCD11_ATTR)	16	R/W	See section	24.3.24/ 1030
1166	TCD Signed Source Address Offset (eDMA_TCD11_SOFF)	16	R/W	See section	24.3.25/ 1031
1168	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD11_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1168	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD11_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1168	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD11_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
116C	TCD Last Source Address Adjustment (eDMA_TCD11_SLAST)	32	R/W	See section	24.3.29/ 1035
1170	TCD Destination Address (eDMA_TCD11_DADDR)	32	R/W	See section	24.3.30/ 1035
1174	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD11_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1174	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD11_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1176	TCD Signed Destination Address Offset (eDMA_TCD11_DOFF)	16	R/W	See section	24.3.33/ 1038
1178	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD11_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
117C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD11_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
117C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD11_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
117E	TCD Control and Status (eDMA_TCD11_CSR)	16	R/W	See section	24.3.37/ 1041
1180	TCD Source Address (eDMA_TCD12_SADDR)	32	R/W	See section	24.3.23/ 1030
1184	TCD Transfer Attributes (eDMA_TCD12_ATTR)	16	R/W	See section	24.3.24/ 1030
1186	TCD Signed Source Address Offset (eDMA_TCD12_SOFF)	16	R/W	See section	24.3.25/ 1031
1188	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD12_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1188	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD12_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1188	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD12_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
118C	TCD Last Source Address Adjustment (eDMA_TCD12_SLAST)	32	R/W	See section	24.3.29/ 1035
1190	TCD Destination Address (eDMA_TCD12_DADDR)	32	R/W	See section	24.3.30/ 1035
1194	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD12_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1194	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD12_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1196	TCD Signed Destination Address Offset (eDMA_TCD12_DOFF)	16	R/W	See section	24.3.33/ 1038
1198	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD12_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
119C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD12_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
119C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD12_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
119E	TCD Control and Status (eDMA_TCD12_CSR)	16	R/W	See section	24.3.37/ 1041
11A0	TCD Source Address (eDMA_TCD13_SADDR)	32	R/W	See section	24.3.23/ 1030
11A4	TCD Transfer Attributes (eDMA_TCD13_ATTR)	16	R/W	See section	24.3.24/ 1030
11A6	TCD Signed Source Address Offset (eDMA_TCD13_SOFF)	16	R/W	See section	24.3.25/ 1031
11A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD13_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
11A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD13_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
11A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD13_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
11AC	TCD Last Source Address Adjustment (eDMA_TCD13_SLAST)	32	R/W	See section	24.3.29/ 1035
11B0	TCD Destination Address (eDMA_TCD13_DADDR)	32	R/W	See section	24.3.30/ 1035
11B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD13_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
11B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD13_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
11B6	TCD Signed Destination Address Offset (eDMA_TCD13_DOFF)	16	R/W	See section	24.3.33/ 1038

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD13_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
11BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD13_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
11BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD13_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
11BE	TCD Control and Status (eDMA_TCD13_CSR)	16	R/W	See section	24.3.37/ 1041
11C0	TCD Source Address (eDMA_TCD14_SADDR)	32	R/W	See section	24.3.23/ 1030
11C4	TCD Transfer Attributes (eDMA_TCD14_ATTR)	16	R/W	See section	24.3.24/ 1030
11C6	TCD Signed Source Address Offset (eDMA_TCD14_SOFF)	16	R/W	See section	24.3.25/ 1031
11C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD14_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
11C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD14_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
11C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD14_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
11CC	TCD Last Source Address Adjustment (eDMA_TCD14_SLAST)	32	R/W	See section	24.3.29/ 1035
11D0	TCD Destination Address (eDMA_TCD14_DADDR)	32	R/W	See section	24.3.30/ 1035
11D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD14_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
11D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD14_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
11D6	TCD Signed Destination Address Offset (eDMA_TCD14_DOFF)	16	R/W	See section	24.3.33/ 1038
11D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD14_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
11DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD14_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
11DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD14_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
11DE	TCD Control and Status (eDMA_TCD14_CSR)	16	R/W	See section	24.3.37/ 1041
11E0	TCD Source Address (eDMA_TCD15_SADDR)	32	R/W	See section	24.3.23/ 1030
11E4	TCD Transfer Attributes (eDMA_TCD15_ATTR)	16	R/W	See section	24.3.24/ 1030
11E6	TCD Signed Source Address Offset (eDMA_TCD15_SOFF)	16	R/W	See section	24.3.25/ 1031

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD15_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
11E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD15_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
11E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD15_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
11EC	TCD Last Source Address Adjustment (eDMA_TCD15_SLAST)	32	R/W	See section	24.3.29/ 1035
11F0	TCD Destination Address (eDMA_TCD15_DADDR)	32	R/W	See section	24.3.30/ 1035
11F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD15_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
11F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD15_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
11F6	TCD Signed Destination Address Offset (eDMA_TCD15_DOFF)	16	R/W	See section	24.3.33/ 1038
11F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD15_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
11FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD15_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
11FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD15_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
11FE	TCD Control and Status (eDMA_TCD15_CSR)	16	R/W	See section	24.3.37/ 1041
1200	TCD Source Address (eDMA_TCD16_SADDR)	32	R/W	See section	24.3.23/ 1030
1204	TCD Transfer Attributes (eDMA_TCD16_ATTR)	16	R/W	See section	24.3.24/ 1030
1206	TCD Signed Source Address Offset (eDMA_TCD16_SOFF)	16	R/W	See section	24.3.25/ 1031
1208	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD16_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1208	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD16_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1208	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD16_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
120C	TCD Last Source Address Adjustment (eDMA_TCD16_SLAST)	32	R/W	See section	24.3.29/ 1035
1210	TCD Destination Address (eDMA_TCD16_DADDR)	32	R/W	See section	24.3.30/ 1035
1214	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD16_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1214	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD16_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1216	TCD Signed Destination Address Offset (eDMA_TCD16_DOFF)	16	R/W	See section	24.3.33/ 1038
1218	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD16_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
121C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD16_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
121C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD16_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
121E	TCD Control and Status (eDMA_TCD16_CSR)	16	R/W	See section	24.3.37/ 1041
1220	TCD Source Address (eDMA_TCD17_SADDR)	32	R/W	See section	24.3.23/ 1030
1224	TCD Transfer Attributes (eDMA_TCD17_ATTR)	16	R/W	See section	24.3.24/ 1030
1226	TCD Signed Source Address Offset (eDMA_TCD17_SOFF)	16	R/W	See section	24.3.25/ 1031
1228	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD17_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1228	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD17_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1228	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD17_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
122C	TCD Last Source Address Adjustment (eDMA_TCD17_SLAST)	32	R/W	See section	24.3.29/ 1035
1230	TCD Destination Address (eDMA_TCD17_DADDR)	32	R/W	See section	24.3.30/ 1035
1234	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD17_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1234	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD17_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1236	TCD Signed Destination Address Offset (eDMA_TCD17_DOFF)	16	R/W	See section	24.3.33/ 1038
1238	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD17_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
123C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD17_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
123C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD17_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
123E	TCD Control and Status (eDMA_TCD17_CSR)	16	R/W	See section	24.3.37/ 1041
1240	TCD Source Address (eDMA_TCD18_SADDR)	32	R/W	See section	24.3.23/ 1030
1244	TCD Transfer Attributes (eDMA_TCD18_ATTR)	16	R/W	See section	24.3.24/ 1030

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1246	TCD Signed Source Address Offset (eDMA_TCD18_SOFF)	16	R/W	See section	24.3.25/ 1031
1248	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD18_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1248	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD18_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1248	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD18_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
124C	TCD Last Source Address Adjustment (eDMA_TCD18_SLAST)	32	R/W	See section	24.3.29/ 1035
1250	TCD Destination Address (eDMA_TCD18_DADDR)	32	R/W	See section	24.3.30/ 1035
1254	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD18_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1254	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD18_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1256	TCD Signed Destination Address Offset (eDMA_TCD18_DOFF)	16	R/W	See section	24.3.33/ 1038
1258	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD18_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
125C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD18_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
125C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD18_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
125E	TCD Control and Status (eDMA_TCD18_CSR)	16	R/W	See section	24.3.37/ 1041
1260	TCD Source Address (eDMA_TCD19_SADDR)	32	R/W	See section	24.3.23/ 1030
1264	TCD Transfer Attributes (eDMA_TCD19_ATTR)	16	R/W	See section	24.3.24/ 1030
1266	TCD Signed Source Address Offset (eDMA_TCD19_SOFF)	16	R/W	See section	24.3.25/ 1031
1268	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD19_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1268	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD19_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1268	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD19_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
126C	TCD Last Source Address Adjustment (eDMA_TCD19_SLAST)	32	R/W	See section	24.3.29/ 1035
1270	TCD Destination Address (eDMA_TCD19_DADDR)	32	R/W	See section	24.3.30/ 1035
1274	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD19_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1274	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD19_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1276	TCD Signed Destination Address Offset (eDMA_TCD19_DOFF)	16	R/W	See section	24.3.33/ 1038
1278	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD19_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
127C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD19_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
127C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD19_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
127E	TCD Control and Status (eDMA_TCD19_CSR)	16	R/W	See section	24.3.37/ 1041
1280	TCD Source Address (eDMA_TCD20_SADDR)	32	R/W	See section	24.3.23/ 1030
1284	TCD Transfer Attributes (eDMA_TCD20_ATTR)	16	R/W	See section	24.3.24/ 1030
1286	TCD Signed Source Address Offset (eDMA_TCD20_SOFF)	16	R/W	See section	24.3.25/ 1031
1288	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD20_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1288	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD20_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1288	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD20_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
128C	TCD Last Source Address Adjustment (eDMA_TCD20_SLAST)	32	R/W	See section	24.3.29/ 1035
1290	TCD Destination Address (eDMA_TCD20_DADDR)	32	R/W	See section	24.3.30/ 1035
1294	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD20_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1294	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD20_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1296	TCD Signed Destination Address Offset (eDMA_TCD20_DOFF)	16	R/W	See section	24.3.33/ 1038
1298	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD20_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
129C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD20_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
129C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD20_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
129E	TCD Control and Status (eDMA_TCD20_CSR)	16	R/W	See section	24.3.37/ 1041
12A0	TCD Source Address (eDMA_TCD21_SADDR)	32	R/W	See section	24.3.23/ 1030

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12A4	TCD Transfer Attributes (eDMA_TCD21_ATTR)	16	R/W	See section	24.3.24/ 1030
12A6	TCD Signed Source Address Offset (eDMA_TCD21_SOFF)	16	R/W	See section	24.3.25/ 1031
12A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD21_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
12A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD21_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
12A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD21_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
12AC	TCD Last Source Address Adjustment (eDMA_TCD21_SLAST)	32	R/W	See section	24.3.29/ 1035
12B0	TCD Destination Address (eDMA_TCD21_DADDR)	32	R/W	See section	24.3.30/ 1035
12B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD21_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
12B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD21_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
12B6	TCD Signed Destination Address Offset (eDMA_TCD21_DOFF)	16	R/W	See section	24.3.33/ 1038
12B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD21_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
12BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD21_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
12BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD21_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
12BE	TCD Control and Status (eDMA_TCD21_CSR)	16	R/W	See section	24.3.37/ 1041
12C0	TCD Source Address (eDMA_TCD22_SADDR)	32	R/W	See section	24.3.23/ 1030
12C4	TCD Transfer Attributes (eDMA_TCD22_ATTR)	16	R/W	See section	24.3.24/ 1030
12C6	TCD Signed Source Address Offset (eDMA_TCD22_SOFF)	16	R/W	See section	24.3.25/ 1031
12C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD22_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
12C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD22_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
12C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD22_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
12CC	TCD Last Source Address Adjustment (eDMA_TCD22_SLAST)	32	R/W	See section	24.3.29/ 1035
12D0	TCD Destination Address (eDMA_TCD22_DADDR)	32	R/W	See section	24.3.30/ 1035

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD22_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
12D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD22_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
12D6	TCD Signed Destination Address Offset (eDMA_TCD22_DOFF)	16	R/W	See section	24.3.33/ 1038
12D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD22_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
12DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD22_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
12DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD22_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
12DE	TCD Control and Status (eDMA_TCD22_CSR)	16	R/W	See section	24.3.37/ 1041
12E0	TCD Source Address (eDMA_TCD23_SADDR)	32	R/W	See section	24.3.23/ 1030
12E4	TCD Transfer Attributes (eDMA_TCD23_ATTR)	16	R/W	See section	24.3.24/ 1030
12E6	TCD Signed Source Address Offset (eDMA_TCD23_SOFF)	16	R/W	See section	24.3.25/ 1031
12E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD23_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
12E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD23_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
12E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD23_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
12EC	TCD Last Source Address Adjustment (eDMA_TCD23_SLAST)	32	R/W	See section	24.3.29/ 1035
12F0	TCD Destination Address (eDMA_TCD23_DADDR)	32	R/W	See section	24.3.30/ 1035
12F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD23_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
12F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD23_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
12F6	TCD Signed Destination Address Offset (eDMA_TCD23_DOFF)	16	R/W	See section	24.3.33/ 1038
12F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD23_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
12FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD23_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
12FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD23_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
12FE	TCD Control and Status (eDMA_TCD23_CSR)	16	R/W	See section	24.3.37/ 1041

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1300	TCD Source Address (eDMA_TCD24_SADDR)	32	R/W	See section	24.3.23/ 1030
1304	TCD Transfer Attributes (eDMA_TCD24_ATTR)	16	R/W	See section	24.3.24/ 1030
1306	TCD Signed Source Address Offset (eDMA_TCD24_SOFF)	16	R/W	See section	24.3.25/ 1031
1308	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD24_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1308	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD24_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1308	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD24_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
130C	TCD Last Source Address Adjustment (eDMA_TCD24_SLAST)	32	R/W	See section	24.3.29/ 1035
1310	TCD Destination Address (eDMA_TCD24_DADDR)	32	R/W	See section	24.3.30/ 1035
1314	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD24_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1314	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD24_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1316	TCD Signed Destination Address Offset (eDMA_TCD24_DOFF)	16	R/W	See section	24.3.33/ 1038
1318	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD24_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
131C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD24_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
131C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD24_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
131E	TCD Control and Status (eDMA_TCD24_CSR)	16	R/W	See section	24.3.37/ 1041
1320	TCD Source Address (eDMA_TCD25_SADDR)	32	R/W	See section	24.3.23/ 1030
1324	TCD Transfer Attributes (eDMA_TCD25_ATTR)	16	R/W	See section	24.3.24/ 1030
1326	TCD Signed Source Address Offset (eDMA_TCD25_SOFF)	16	R/W	See section	24.3.25/ 1031
1328	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD25_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1328	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD25_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1328	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD25_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
132C	TCD Last Source Address Adjustment (eDMA_TCD25_SLAST)	32	R/W	See section	24.3.29/ 1035

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1330	TCD Destination Address (eDMA_TCD25_DADDR)	32	R/W	See section	24.3.30/ 1035
1334	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD25_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1334	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD25_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1336	TCD Signed Destination Address Offset (eDMA_TCD25_DOFF)	16	R/W	See section	24.3.33/ 1038
1338	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD25_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
133C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD25_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
133C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD25_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
133E	TCD Control and Status (eDMA_TCD25_CSR)	16	R/W	See section	24.3.37/ 1041
1340	TCD Source Address (eDMA_TCD26_SADDR)	32	R/W	See section	24.3.23/ 1030
1344	TCD Transfer Attributes (eDMA_TCD26_ATTR)	16	R/W	See section	24.3.24/ 1030
1346	TCD Signed Source Address Offset (eDMA_TCD26_SOFF)	16	R/W	See section	24.3.25/ 1031
1348	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD26_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1348	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD26_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1348	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD26_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
134C	TCD Last Source Address Adjustment (eDMA_TCD26_SLAST)	32	R/W	See section	24.3.29/ 1035
1350	TCD Destination Address (eDMA_TCD26_DADDR)	32	R/W	See section	24.3.30/ 1035
1354	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD26_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1354	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD26_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1356	TCD Signed Destination Address Offset (eDMA_TCD26_DOFF)	16	R/W	See section	24.3.33/ 1038
1358	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD26_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
135C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD26_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
135C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD26_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
135E	TCD Control and Status (eDMA_TCD26_CSR)	16	R/W	See section	24.3.37/ 1041
1360	TCD Source Address (eDMA_TCD27_SADDR)	32	R/W	See section	24.3.23/ 1030
1364	TCD Transfer Attributes (eDMA_TCD27_ATTR)	16	R/W	See section	24.3.24/ 1030
1366	TCD Signed Source Address Offset (eDMA_TCD27_SOFF)	16	R/W	See section	24.3.25/ 1031
1368	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD27_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1368	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD27_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1368	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD27_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
136C	TCD Last Source Address Adjustment (eDMA_TCD27_SLAST)	32	R/W	See section	24.3.29/ 1035
1370	TCD Destination Address (eDMA_TCD27_DADDR)	32	R/W	See section	24.3.30/ 1035
1374	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD27_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1374	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD27_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1376	TCD Signed Destination Address Offset (eDMA_TCD27_DOFF)	16	R/W	See section	24.3.33/ 1038
1378	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD27_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
137C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD27_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
137C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD27_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
137E	TCD Control and Status (eDMA_TCD27_CSR)	16	R/W	See section	24.3.37/ 1041
1380	TCD Source Address (eDMA_TCD28_SADDR)	32	R/W	See section	24.3.23/ 1030
1384	TCD Transfer Attributes (eDMA_TCD28_ATTR)	16	R/W	See section	24.3.24/ 1030
1386	TCD Signed Source Address Offset (eDMA_TCD28_SOFF)	16	R/W	See section	24.3.25/ 1031
1388	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD28_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1388	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD28_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1388	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD28_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
138C	TCD Last Source Address Adjustment (eDMA_TCD28_SLAST)	32	R/W	See section	24.3.29/ 1035
1390	TCD Destination Address (eDMA_TCD28_DADDR)	32	R/W	See section	24.3.30/ 1035
1394	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD28_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1394	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD28_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1396	TCD Signed Destination Address Offset (eDMA_TCD28_DOFF)	16	R/W	See section	24.3.33/ 1038
1398	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD28_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
139C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD28_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
139C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD28_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
139E	TCD Control and Status (eDMA_TCD28_CSR)	16	R/W	See section	24.3.37/ 1041
13A0	TCD Source Address (eDMA_TCD29_SADDR)	32	R/W	See section	24.3.23/ 1030
13A4	TCD Transfer Attributes (eDMA_TCD29_ATTR)	16	R/W	See section	24.3.24/ 1030
13A6	TCD Signed Source Address Offset (eDMA_TCD29_SOFF)	16	R/W	See section	24.3.25/ 1031
13A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD29_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
13A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD29_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
13A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD29_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
13AC	TCD Last Source Address Adjustment (eDMA_TCD29_SLAST)	32	R/W	See section	24.3.29/ 1035
13B0	TCD Destination Address (eDMA_TCD29_DADDR)	32	R/W	See section	24.3.30/ 1035
13B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD29_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
13B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD29_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
13B6	TCD Signed Destination Address Offset (eDMA_TCD29_DOFF)	16	R/W	See section	24.3.33/ 1038
13B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD29_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
13BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD29_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD29_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
13BE	TCD Control and Status (eDMA_TCD29_CSR)	16	R/W	See section	24.3.37/ 1041
13C0	TCD Source Address (eDMA_TCD30_SADDR)	32	R/W	See section	24.3.23/ 1030
13C4	TCD Transfer Attributes (eDMA_TCD30_ATTR)	16	R/W	See section	24.3.24/ 1030
13C6	TCD Signed Source Address Offset (eDMA_TCD30_SOFF)	16	R/W	See section	24.3.25/ 1031
13C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD30_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
13C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD30_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
13C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD30_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
13CC	TCD Last Source Address Adjustment (eDMA_TCD30_SLAST)	32	R/W	See section	24.3.29/ 1035
13D0	TCD Destination Address (eDMA_TCD30_DADDR)	32	R/W	See section	24.3.30/ 1035
13D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD30_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
13D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD30_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
13D6	TCD Signed Destination Address Offset (eDMA_TCD30_DOFF)	16	R/W	See section	24.3.33/ 1038
13D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD30_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
13DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD30_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
13DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD30_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
13DE	TCD Control and Status (eDMA_TCD30_CSR)	16	R/W	See section	24.3.37/ 1041
13E0	TCD Source Address (eDMA_TCD31_SADDR)	32	R/W	See section	24.3.23/ 1030
13E4	TCD Transfer Attributes (eDMA_TCD31_ATTR)	16	R/W	See section	24.3.24/ 1030
13E6	TCD Signed Source Address Offset (eDMA_TCD31_SOFF)	16	R/W	See section	24.3.25/ 1031
13E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD31_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
13E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD31_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD31_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
13EC	TCD Last Source Address Adjustment (eDMA_TCD31_SLAST)	32	R/W	See section	24.3.29/ 1035
13F0	TCD Destination Address (eDMA_TCD31_DADDR)	32	R/W	See section	24.3.30/ 1035
13F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD31_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
13F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD31_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
13F6	TCD Signed Destination Address Offset (eDMA_TCD31_DOFF)	16	R/W	See section	24.3.33/ 1038
13F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD31_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
13FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD31_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
13FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD31_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
13FE	TCD Control and Status (eDMA_TCD31_CSR)	16	R/W	See section	24.3.37/ 1041
1400	TCD Source Address (eDMA_TCD32_SADDR)	32	R/W	See section	24.3.23/ 1030
1404	TCD Transfer Attributes (eDMA_TCD32_ATTR)	16	R/W	See section	24.3.24/ 1030
1406	TCD Signed Source Address Offset (eDMA_TCD32_SOFF)	16	R/W	See section	24.3.25/ 1031
1408	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD32_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1408	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD32_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1408	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD32_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
140C	TCD Last Source Address Adjustment (eDMA_TCD32_SLAST)	32	R/W	See section	24.3.29/ 1035
1410	TCD Destination Address (eDMA_TCD32_DADDR)	32	R/W	See section	24.3.30/ 1035
1414	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD32_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1414	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD32_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1416	TCD Signed Destination Address Offset (eDMA_TCD32_DOFF)	16	R/W	See section	24.3.33/ 1038
1418	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD32_DLASTSGA)	32	R/W	See section	24.3.34/ 1038

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
141C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD32_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
141C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD32_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
141E	TCD Control and Status (eDMA_TCD32_CSR)	16	R/W	See section	24.3.37/ 1041
1420	TCD Source Address (eDMA_TCD33_SADDR)	32	R/W	See section	24.3.23/ 1030
1424	TCD Transfer Attributes (eDMA_TCD33_ATTR)	16	R/W	See section	24.3.24/ 1030
1426	TCD Signed Source Address Offset (eDMA_TCD33_SOFF)	16	R/W	See section	24.3.25/ 1031
1428	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD33_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1428	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD33_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1428	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD33_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
142C	TCD Last Source Address Adjustment (eDMA_TCD33_SLAST)	32	R/W	See section	24.3.29/ 1035
1430	TCD Destination Address (eDMA_TCD33_DADDR)	32	R/W	See section	24.3.30/ 1035
1434	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD33_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1434	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD33_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1436	TCD Signed Destination Address Offset (eDMA_TCD33_DOFF)	16	R/W	See section	24.3.33/ 1038
1438	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD33_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
143C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD33_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
143C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD33_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
143E	TCD Control and Status (eDMA_TCD33_CSR)	16	R/W	See section	24.3.37/ 1041
1440	TCD Source Address (eDMA_TCD34_SADDR)	32	R/W	See section	24.3.23/ 1030
1444	TCD Transfer Attributes (eDMA_TCD34_ATTR)	16	R/W	See section	24.3.24/ 1030
1446	TCD Signed Source Address Offset (eDMA_TCD34_SOFF)	16	R/W	See section	24.3.25/ 1031
1448	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD34_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1448	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD34_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1448	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD34_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
144C	TCD Last Source Address Adjustment (eDMA_TCD34_SLAST)	32	R/W	See section	24.3.29/ 1035
1450	TCD Destination Address (eDMA_TCD34_DADDR)	32	R/W	See section	24.3.30/ 1035
1454	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD34_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1454	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD34_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1456	TCD Signed Destination Address Offset (eDMA_TCD34_DOFF)	16	R/W	See section	24.3.33/ 1038
1458	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD34_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
145C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD34_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
145C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD34_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
145E	TCD Control and Status (eDMA_TCD34_CSR)	16	R/W	See section	24.3.37/ 1041
1460	TCD Source Address (eDMA_TCD35_SADDR)	32	R/W	See section	24.3.23/ 1030
1464	TCD Transfer Attributes (eDMA_TCD35_ATTR)	16	R/W	See section	24.3.24/ 1030
1466	TCD Signed Source Address Offset (eDMA_TCD35_SOFF)	16	R/W	See section	24.3.25/ 1031
1468	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD35_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1468	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD35_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1468	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD35_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
146C	TCD Last Source Address Adjustment (eDMA_TCD35_SLAST)	32	R/W	See section	24.3.29/ 1035
1470	TCD Destination Address (eDMA_TCD35_DADDR)	32	R/W	See section	24.3.30/ 1035
1474	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD35_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1474	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD35_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1476	TCD Signed Destination Address Offset (eDMA_TCD35_DOFF)	16	R/W	See section	24.3.33/ 1038

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1478	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD35_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
147C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD35_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
147C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD35_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
147E	TCD Control and Status (eDMA_TCD35_CSR)	16	R/W	See section	24.3.37/ 1041
1480	TCD Source Address (eDMA_TCD36_SADDR)	32	R/W	See section	24.3.23/ 1030
1484	TCD Transfer Attributes (eDMA_TCD36_ATTR)	16	R/W	See section	24.3.24/ 1030
1486	TCD Signed Source Address Offset (eDMA_TCD36_SOFF)	16	R/W	See section	24.3.25/ 1031
1488	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD36_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1488	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD36_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1488	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD36_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
148C	TCD Last Source Address Adjustment (eDMA_TCD36_SLAST)	32	R/W	See section	24.3.29/ 1035
1490	TCD Destination Address (eDMA_TCD36_DADDR)	32	R/W	See section	24.3.30/ 1035
1494	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD36_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1494	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD36_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1496	TCD Signed Destination Address Offset (eDMA_TCD36_DOFF)	16	R/W	See section	24.3.33/ 1038
1498	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD36_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
149C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD36_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
149C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD36_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
149E	TCD Control and Status (eDMA_TCD36_CSR)	16	R/W	See section	24.3.37/ 1041
14A0	TCD Source Address (eDMA_TCD37_SADDR)	32	R/W	See section	24.3.23/ 1030
14A4	TCD Transfer Attributes (eDMA_TCD37_ATTR)	16	R/W	See section	24.3.24/ 1030
14A6	TCD Signed Source Address Offset (eDMA_TCD37_SOFF)	16	R/W	See section	24.3.25/ 1031

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD37_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
14A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD37_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
14A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD37_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
14AC	TCD Last Source Address Adjustment (eDMA_TCD37_SLAST)	32	R/W	See section	24.3.29/ 1035
14B0	TCD Destination Address (eDMA_TCD37_DADDR)	32	R/W	See section	24.3.30/ 1035
14B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD37_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
14B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD37_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
14B6	TCD Signed Destination Address Offset (eDMA_TCD37_DOFF)	16	R/W	See section	24.3.33/ 1038
14B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD37_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
14BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD37_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
14BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD37_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
14BE	TCD Control and Status (eDMA_TCD37_CSR)	16	R/W	See section	24.3.37/ 1041
14C0	TCD Source Address (eDMA_TCD38_SADDR)	32	R/W	See section	24.3.23/ 1030
14C4	TCD Transfer Attributes (eDMA_TCD38_ATTR)	16	R/W	See section	24.3.24/ 1030
14C6	TCD Signed Source Address Offset (eDMA_TCD38_SOFF)	16	R/W	See section	24.3.25/ 1031
14C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD38_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
14C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD38_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
14C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD38_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
14CC	TCD Last Source Address Adjustment (eDMA_TCD38_SLAST)	32	R/W	See section	24.3.29/ 1035
14D0	TCD Destination Address (eDMA_TCD38_DADDR)	32	R/W	See section	24.3.30/ 1035
14D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD38_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
14D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD38_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14D6	TCD Signed Destination Address Offset (eDMA_TCD38_DOFF)	16	R/W	See section	24.3.33/ 1038
14D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD38_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
14DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD38_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
14DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD38_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
14DE	TCD Control and Status (eDMA_TCD38_CSR)	16	R/W	See section	24.3.37/ 1041
14E0	TCD Source Address (eDMA_TCD39_SADDR)	32	R/W	See section	24.3.23/ 1030
14E4	TCD Transfer Attributes (eDMA_TCD39_ATTR)	16	R/W	See section	24.3.24/ 1030
14E6	TCD Signed Source Address Offset (eDMA_TCD39_SOFF)	16	R/W	See section	24.3.25/ 1031
14E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD39_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
14E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD39_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
14E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD39_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
14EC	TCD Last Source Address Adjustment (eDMA_TCD39_SLAST)	32	R/W	See section	24.3.29/ 1035
14F0	TCD Destination Address (eDMA_TCD39_DADDR)	32	R/W	See section	24.3.30/ 1035
14F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD39_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
14F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD39_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
14F6	TCD Signed Destination Address Offset (eDMA_TCD39_DOFF)	16	R/W	See section	24.3.33/ 1038
14F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD39_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
14FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD39_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
14FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD39_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
14FE	TCD Control and Status (eDMA_TCD39_CSR)	16	R/W	See section	24.3.37/ 1041
1500	TCD Source Address (eDMA_TCD40_SADDR)	32	R/W	See section	24.3.23/ 1030
1504	TCD Transfer Attributes (eDMA_TCD40_ATTR)	16	R/W	See section	24.3.24/ 1030

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1506	TCD Signed Source Address Offset (eDMA_TCD40_SOFF)	16	R/W	See section	24.3.25/ 1031
1508	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD40_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1508	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD40_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1508	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD40_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
150C	TCD Last Source Address Adjustment (eDMA_TCD40_SLAST)	32	R/W	See section	24.3.29/ 1035
1510	TCD Destination Address (eDMA_TCD40_DADDR)	32	R/W	See section	24.3.30/ 1035
1514	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD40_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1514	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD40_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1516	TCD Signed Destination Address Offset (eDMA_TCD40_DOFF)	16	R/W	See section	24.3.33/ 1038
1518	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD40_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
151C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD40_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
151C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD40_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
151E	TCD Control and Status (eDMA_TCD40_CSR)	16	R/W	See section	24.3.37/ 1041
1520	TCD Source Address (eDMA_TCD41_SADDR)	32	R/W	See section	24.3.23/ 1030
1524	TCD Transfer Attributes (eDMA_TCD41_ATTR)	16	R/W	See section	24.3.24/ 1030
1526	TCD Signed Source Address Offset (eDMA_TCD41_SOFF)	16	R/W	See section	24.3.25/ 1031
1528	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD41_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1528	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD41_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1528	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD41_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
152C	TCD Last Source Address Adjustment (eDMA_TCD41_SLAST)	32	R/W	See section	24.3.29/ 1035
1530	TCD Destination Address (eDMA_TCD41_DADDR)	32	R/W	See section	24.3.30/ 1035
1534	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD41_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1534	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD41_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1536	TCD Signed Destination Address Offset (eDMA_TCD41_DOFF)	16	R/W	See section	24.3.33/ 1038
1538	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD41_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
153C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD41_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
153C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD41_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
153E	TCD Control and Status (eDMA_TCD41_CSR)	16	R/W	See section	24.3.37/ 1041
1540	TCD Source Address (eDMA_TCD42_SADDR)	32	R/W	See section	24.3.23/ 1030
1544	TCD Transfer Attributes (eDMA_TCD42_ATTR)	16	R/W	See section	24.3.24/ 1030
1546	TCD Signed Source Address Offset (eDMA_TCD42_SOFF)	16	R/W	See section	24.3.25/ 1031
1548	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD42_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1548	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD42_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1548	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD42_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
154C	TCD Last Source Address Adjustment (eDMA_TCD42_SLAST)	32	R/W	See section	24.3.29/ 1035
1550	TCD Destination Address (eDMA_TCD42_DADDR)	32	R/W	See section	24.3.30/ 1035
1554	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD42_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1554	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD42_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1556	TCD Signed Destination Address Offset (eDMA_TCD42_DOFF)	16	R/W	See section	24.3.33/ 1038
1558	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD42_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
155C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD42_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
155C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD42_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
155E	TCD Control and Status (eDMA_TCD42_CSR)	16	R/W	See section	24.3.37/ 1041
1560	TCD Source Address (eDMA_TCD43_SADDR)	32	R/W	See section	24.3.23/ 1030

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1564	TCD Transfer Attributes (eDMA_TCD43_ATTR)	16	R/W	See section	24.3.24/ 1030
1566	TCD Signed Source Address Offset (eDMA_TCD43_SOFF)	16	R/W	See section	24.3.25/ 1031
1568	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD43_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1568	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD43_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1568	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD43_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
156C	TCD Last Source Address Adjustment (eDMA_TCD43_SLAST)	32	R/W	See section	24.3.29/ 1035
1570	TCD Destination Address (eDMA_TCD43_DADDR)	32	R/W	See section	24.3.30/ 1035
1574	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD43_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1574	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD43_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1576	TCD Signed Destination Address Offset (eDMA_TCD43_DOFF)	16	R/W	See section	24.3.33/ 1038
1578	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD43_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
157C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD43_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
157C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD43_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
157E	TCD Control and Status (eDMA_TCD43_CSR)	16	R/W	See section	24.3.37/ 1041
1580	TCD Source Address (eDMA_TCD44_SADDR)	32	R/W	See section	24.3.23/ 1030
1584	TCD Transfer Attributes (eDMA_TCD44_ATTR)	16	R/W	See section	24.3.24/ 1030
1586	TCD Signed Source Address Offset (eDMA_TCD44_SOFF)	16	R/W	See section	24.3.25/ 1031
1588	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD44_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1588	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD44_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1588	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD44_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
158C	TCD Last Source Address Adjustment (eDMA_TCD44_SLAST)	32	R/W	See section	24.3.29/ 1035
1590	TCD Destination Address (eDMA_TCD44_DADDR)	32	R/W	See section	24.3.30/ 1035

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1594	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD44_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1594	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD44_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1596	TCD Signed Destination Address Offset (eDMA_TCD44_DOFF)	16	R/W	See section	24.3.33/ 1038
1598	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD44_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
159C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD44_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
159C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD44_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
159E	TCD Control and Status (eDMA_TCD44_CSR)	16	R/W	See section	24.3.37/ 1041
15A0	TCD Source Address (eDMA_TCD45_SADDR)	32	R/W	See section	24.3.23/ 1030
15A4	TCD Transfer Attributes (eDMA_TCD45_ATTR)	16	R/W	See section	24.3.24/ 1030
15A6	TCD Signed Source Address Offset (eDMA_TCD45_SOFF)	16	R/W	See section	24.3.25/ 1031
15A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD45_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
15A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD45_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
15A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD45_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
15AC	TCD Last Source Address Adjustment (eDMA_TCD45_SLAST)	32	R/W	See section	24.3.29/ 1035
15B0	TCD Destination Address (eDMA_TCD45_DADDR)	32	R/W	See section	24.3.30/ 1035
15B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD45_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
15B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD45_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
15B6	TCD Signed Destination Address Offset (eDMA_TCD45_DOFF)	16	R/W	See section	24.3.33/ 1038
15B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD45_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
15BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD45_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
15BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD45_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
15BE	TCD Control and Status (eDMA_TCD45_CSR)	16	R/W	See section	24.3.37/ 1041

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15C0	TCD Source Address (eDMA_TCD46_SADDR)	32	R/W	See section	24.3.23/ 1030
15C4	TCD Transfer Attributes (eDMA_TCD46_ATTR)	16	R/W	See section	24.3.24/ 1030
15C6	TCD Signed Source Address Offset (eDMA_TCD46_SOFF)	16	R/W	See section	24.3.25/ 1031
15C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD46_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
15C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD46_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
15C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD46_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
15CC	TCD Last Source Address Adjustment (eDMA_TCD46_SLAST)	32	R/W	See section	24.3.29/ 1035
15D0	TCD Destination Address (eDMA_TCD46_DADDR)	32	R/W	See section	24.3.30/ 1035
15D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD46_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
15D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD46_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
15D6	TCD Signed Destination Address Offset (eDMA_TCD46_DOFF)	16	R/W	See section	24.3.33/ 1038
15D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD46_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
15DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD46_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
15DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD46_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
15DE	TCD Control and Status (eDMA_TCD46_CSR)	16	R/W	See section	24.3.37/ 1041
15E0	TCD Source Address (eDMA_TCD47_SADDR)	32	R/W	See section	24.3.23/ 1030
15E4	TCD Transfer Attributes (eDMA_TCD47_ATTR)	16	R/W	See section	24.3.24/ 1030
15E6	TCD Signed Source Address Offset (eDMA_TCD47_SOFF)	16	R/W	See section	24.3.25/ 1031
15E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD47_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
15E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD47_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
15E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD47_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
15EC	TCD Last Source Address Adjustment (eDMA_TCD47_SLAST)	32	R/W	See section	24.3.29/ 1035

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15F0	TCD Destination Address (eDMA_TCD47_DADDR)	32	R/W	See section	24.3.30/ 1035
15F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD47_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
15F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD47_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
15F6	TCD Signed Destination Address Offset (eDMA_TCD47_DOFF)	16	R/W	See section	24.3.33/ 1038
15F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD47_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
15FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD47_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
15FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD47_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
15FE	TCD Control and Status (eDMA_TCD47_CSR)	16	R/W	See section	24.3.37/ 1041
1600	TCD Source Address (eDMA_TCD48_SADDR)	32	R/W	See section	24.3.23/ 1030
1604	TCD Transfer Attributes (eDMA_TCD48_ATTR)	16	R/W	See section	24.3.24/ 1030
1606	TCD Signed Source Address Offset (eDMA_TCD48_SOFF)	16	R/W	See section	24.3.25/ 1031
1608	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD48_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1608	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD48_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1608	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD48_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
160C	TCD Last Source Address Adjustment (eDMA_TCD48_SLAST)	32	R/W	See section	24.3.29/ 1035
1610	TCD Destination Address (eDMA_TCD48_DADDR)	32	R/W	See section	24.3.30/ 1035
1614	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD48_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1614	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD48_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1616	TCD Signed Destination Address Offset (eDMA_TCD48_DOFF)	16	R/W	See section	24.3.33/ 1038
1618	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD48_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
161C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD48_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
161C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD48_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
161E	TCD Control and Status (eDMA_TCD48_CSR)	16	R/W	See section	24.3.37/ 1041
1620	TCD Source Address (eDMA_TCD49_SADDR)	32	R/W	See section	24.3.23/ 1030
1624	TCD Transfer Attributes (eDMA_TCD49_ATTR)	16	R/W	See section	24.3.24/ 1030
1626	TCD Signed Source Address Offset (eDMA_TCD49_SOFF)	16	R/W	See section	24.3.25/ 1031
1628	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD49_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1628	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD49_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1628	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD49_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
162C	TCD Last Source Address Adjustment (eDMA_TCD49_SLAST)	32	R/W	See section	24.3.29/ 1035
1630	TCD Destination Address (eDMA_TCD49_DADDR)	32	R/W	See section	24.3.30/ 1035
1634	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD49_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1634	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD49_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1636	TCD Signed Destination Address Offset (eDMA_TCD49_DOFF)	16	R/W	See section	24.3.33/ 1038
1638	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD49_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
163C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD49_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
163C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD49_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
163E	TCD Control and Status (eDMA_TCD49_CSR)	16	R/W	See section	24.3.37/ 1041
1640	TCD Source Address (eDMA_TCD50_SADDR)	32	R/W	See section	24.3.23/ 1030
1644	TCD Transfer Attributes (eDMA_TCD50_ATTR)	16	R/W	See section	24.3.24/ 1030
1646	TCD Signed Source Address Offset (eDMA_TCD50_SOFF)	16	R/W	See section	24.3.25/ 1031
1648	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD50_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1648	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD50_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1648	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD50_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
164C	TCD Last Source Address Adjustment (eDMA_TCD50_SLAST)	32	R/W	See section	24.3.29/ 1035
1650	TCD Destination Address (eDMA_TCD50_DADDR)	32	R/W	See section	24.3.30/ 1035
1654	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD50_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1654	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD50_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1656	TCD Signed Destination Address Offset (eDMA_TCD50_DOFF)	16	R/W	See section	24.3.33/ 1038
1658	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD50_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
165C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD50_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
165C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD50_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
165E	TCD Control and Status (eDMA_TCD50_CSR)	16	R/W	See section	24.3.37/ 1041
1660	TCD Source Address (eDMA_TCD51_SADDR)	32	R/W	See section	24.3.23/ 1030
1664	TCD Transfer Attributes (eDMA_TCD51_ATTR)	16	R/W	See section	24.3.24/ 1030
1666	TCD Signed Source Address Offset (eDMA_TCD51_SOFF)	16	R/W	See section	24.3.25/ 1031
1668	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD51_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1668	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD51_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1668	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD51_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
166C	TCD Last Source Address Adjustment (eDMA_TCD51_SLAST)	32	R/W	See section	24.3.29/ 1035
1670	TCD Destination Address (eDMA_TCD51_DADDR)	32	R/W	See section	24.3.30/ 1035
1674	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD51_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1674	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD51_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1676	TCD Signed Destination Address Offset (eDMA_TCD51_DOFF)	16	R/W	See section	24.3.33/ 1038
1678	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD51_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
167C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD51_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
167C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD51_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
167E	TCD Control and Status (eDMA_TCD51_CSR)	16	R/W	See section	24.3.37/ 1041
1680	TCD Source Address (eDMA_TCD52_SADDR)	32	R/W	See section	24.3.23/ 1030
1684	TCD Transfer Attributes (eDMA_TCD52_ATTR)	16	R/W	See section	24.3.24/ 1030
1686	TCD Signed Source Address Offset (eDMA_TCD52_SOFF)	16	R/W	See section	24.3.25/ 1031
1688	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD52_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1688	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD52_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1688	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD52_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
168C	TCD Last Source Address Adjustment (eDMA_TCD52_SLAST)	32	R/W	See section	24.3.29/ 1035
1690	TCD Destination Address (eDMA_TCD52_DADDR)	32	R/W	See section	24.3.30/ 1035
1694	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD52_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1694	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD52_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1696	TCD Signed Destination Address Offset (eDMA_TCD52_DOFF)	16	R/W	See section	24.3.33/ 1038
1698	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD52_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
169C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD52_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
169C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD52_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
169E	TCD Control and Status (eDMA_TCD52_CSR)	16	R/W	See section	24.3.37/ 1041
16A0	TCD Source Address (eDMA_TCD53_SADDR)	32	R/W	See section	24.3.23/ 1030
16A4	TCD Transfer Attributes (eDMA_TCD53_ATTR)	16	R/W	See section	24.3.24/ 1030
16A6	TCD Signed Source Address Offset (eDMA_TCD53_SOFF)	16	R/W	See section	24.3.25/ 1031
16A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD53_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
16A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD53_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD53_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
16AC	TCD Last Source Address Adjustment (eDMA_TCD53_SLAST)	32	R/W	See section	24.3.29/ 1035
16B0	TCD Destination Address (eDMA_TCD53_DADDR)	32	R/W	See section	24.3.30/ 1035
16B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD53_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
16B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD53_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
16B6	TCD Signed Destination Address Offset (eDMA_TCD53_DOFF)	16	R/W	See section	24.3.33/ 1038
16B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD53_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
16BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD53_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
16BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD53_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
16BE	TCD Control and Status (eDMA_TCD53_CSR)	16	R/W	See section	24.3.37/ 1041
16C0	TCD Source Address (eDMA_TCD54_SADDR)	32	R/W	See section	24.3.23/ 1030
16C4	TCD Transfer Attributes (eDMA_TCD54_ATTR)	16	R/W	See section	24.3.24/ 1030
16C6	TCD Signed Source Address Offset (eDMA_TCD54_SOFF)	16	R/W	See section	24.3.25/ 1031
16C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD54_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
16C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD54_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
16C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD54_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
16CC	TCD Last Source Address Adjustment (eDMA_TCD54_SLAST)	32	R/W	See section	24.3.29/ 1035
16D0	TCD Destination Address (eDMA_TCD54_DADDR)	32	R/W	See section	24.3.30/ 1035
16D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD54_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
16D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD54_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
16D6	TCD Signed Destination Address Offset (eDMA_TCD54_DOFF)	16	R/W	See section	24.3.33/ 1038
16D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD54_DLASTSGA)	32	R/W	See section	24.3.34/ 1038

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD54_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
16DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD54_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
16DE	TCD Control and Status (eDMA_TCD54_CSR)	16	R/W	See section	24.3.37/ 1041
16E0	TCD Source Address (eDMA_TCD55_SADDR)	32	R/W	See section	24.3.23/ 1030
16E4	TCD Transfer Attributes (eDMA_TCD55_ATTR)	16	R/W	See section	24.3.24/ 1030
16E6	TCD Signed Source Address Offset (eDMA_TCD55_SOFF)	16	R/W	See section	24.3.25/ 1031
16E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD55_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
16E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD55_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
16E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD55_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
16EC	TCD Last Source Address Adjustment (eDMA_TCD55_SLAST)	32	R/W	See section	24.3.29/ 1035
16F0	TCD Destination Address (eDMA_TCD55_DADDR)	32	R/W	See section	24.3.30/ 1035
16F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD55_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
16F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD55_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
16F6	TCD Signed Destination Address Offset (eDMA_TCD55_DOFF)	16	R/W	See section	24.3.33/ 1038
16F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD55_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
16FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD55_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
16FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD55_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
16FE	TCD Control and Status (eDMA_TCD55_CSR)	16	R/W	See section	24.3.37/ 1041
1700	TCD Source Address (eDMA_TCD56_SADDR)	32	R/W	See section	24.3.23/ 1030
1704	TCD Transfer Attributes (eDMA_TCD56_ATTR)	16	R/W	See section	24.3.24/ 1030
1706	TCD Signed Source Address Offset (eDMA_TCD56_SOFF)	16	R/W	See section	24.3.25/ 1031
1708	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD56_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1708	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD56_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1708	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD56_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
170C	TCD Last Source Address Adjustment (eDMA_TCD56_SLAST)	32	R/W	See section	24.3.29/ 1035
1710	TCD Destination Address (eDMA_TCD56_DADDR)	32	R/W	See section	24.3.30/ 1035
1714	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD56_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1714	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD56_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1716	TCD Signed Destination Address Offset (eDMA_TCD56_DOFF)	16	R/W	See section	24.3.33/ 1038
1718	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD56_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
171C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD56_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
171C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD56_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
171E	TCD Control and Status (eDMA_TCD56_CSR)	16	R/W	See section	24.3.37/ 1041
1720	TCD Source Address (eDMA_TCD57_SADDR)	32	R/W	See section	24.3.23/ 1030
1724	TCD Transfer Attributes (eDMA_TCD57_ATTR)	16	R/W	See section	24.3.24/ 1030
1726	TCD Signed Source Address Offset (eDMA_TCD57_SOFF)	16	R/W	See section	24.3.25/ 1031
1728	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD57_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1728	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD57_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1728	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD57_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
172C	TCD Last Source Address Adjustment (eDMA_TCD57_SLAST)	32	R/W	See section	24.3.29/ 1035
1730	TCD Destination Address (eDMA_TCD57_DADDR)	32	R/W	See section	24.3.30/ 1035
1734	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD57_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1734	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD57_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1736	TCD Signed Destination Address Offset (eDMA_TCD57_DOFF)	16	R/W	See section	24.3.33/ 1038

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1738	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD57_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
173C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD57_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
173C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD57_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
173E	TCD Control and Status (eDMA_TCD57_CSR)	16	R/W	See section	24.3.37/ 1041
1740	TCD Source Address (eDMA_TCD58_SADDR)	32	R/W	See section	24.3.23/ 1030
1744	TCD Transfer Attributes (eDMA_TCD58_ATTR)	16	R/W	See section	24.3.24/ 1030
1746	TCD Signed Source Address Offset (eDMA_TCD58_SOFF)	16	R/W	See section	24.3.25/ 1031
1748	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD58_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1748	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD58_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1748	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD58_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
174C	TCD Last Source Address Adjustment (eDMA_TCD58_SLAST)	32	R/W	See section	24.3.29/ 1035
1750	TCD Destination Address (eDMA_TCD58_DADDR)	32	R/W	See section	24.3.30/ 1035
1754	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD58_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1754	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD58_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1756	TCD Signed Destination Address Offset (eDMA_TCD58_DOFF)	16	R/W	See section	24.3.33/ 1038
1758	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD58_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
175C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD58_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
175C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD58_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
175E	TCD Control and Status (eDMA_TCD58_CSR)	16	R/W	See section	24.3.37/ 1041
1760	TCD Source Address (eDMA_TCD59_SADDR)	32	R/W	See section	24.3.23/ 1030
1764	TCD Transfer Attributes (eDMA_TCD59_ATTR)	16	R/W	See section	24.3.24/ 1030
1766	TCD Signed Source Address Offset (eDMA_TCD59_SOFF)	16	R/W	See section	24.3.25/ 1031

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1768	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD59_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1768	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD59_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1768	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD59_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
176C	TCD Last Source Address Adjustment (eDMA_TCD59_SLAST)	32	R/W	See section	24.3.29/ 1035
1770	TCD Destination Address (eDMA_TCD59_DADDR)	32	R/W	See section	24.3.30/ 1035
1774	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD59_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1774	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD59_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
1776	TCD Signed Destination Address Offset (eDMA_TCD59_DOFF)	16	R/W	See section	24.3.33/ 1038
1778	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD59_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
177C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD59_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
177C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD59_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
177E	TCD Control and Status (eDMA_TCD59_CSR)	16	R/W	See section	24.3.37/ 1041
1780	TCD Source Address (eDMA_TCD60_SADDR)	32	R/W	See section	24.3.23/ 1030
1784	TCD Transfer Attributes (eDMA_TCD60_ATTR)	16	R/W	See section	24.3.24/ 1030
1786	TCD Signed Source Address Offset (eDMA_TCD60_SOFF)	16	R/W	See section	24.3.25/ 1031
1788	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD60_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
1788	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD60_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
1788	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD60_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
178C	TCD Last Source Address Adjustment (eDMA_TCD60_SLAST)	32	R/W	See section	24.3.29/ 1035
1790	TCD Destination Address (eDMA_TCD60_DADDR)	32	R/W	See section	24.3.30/ 1035
1794	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD60_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
1794	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD60_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1796	TCD Signed Destination Address Offset (eDMA_TCD60_DOFF)	16	R/W	See section	24.3.33/ 1038
1798	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD60_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
179C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD60_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
179C	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD60_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
179E	TCD Control and Status (eDMA_TCD60_CSR)	16	R/W	See section	24.3.37/ 1041
17A0	TCD Source Address (eDMA_TCD61_SADDR)	32	R/W	See section	24.3.23/ 1030
17A4	TCD Transfer Attributes (eDMA_TCD61_ATTR)	16	R/W	See section	24.3.24/ 1030
17A6	TCD Signed Source Address Offset (eDMA_TCD61_SOFF)	16	R/W	See section	24.3.25/ 1031
17A8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD61_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
17A8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD61_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
17A8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD61_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
17AC	TCD Last Source Address Adjustment (eDMA_TCD61_SLAST)	32	R/W	See section	24.3.29/ 1035
17B0	TCD Destination Address (eDMA_TCD61_DADDR)	32	R/W	See section	24.3.30/ 1035
17B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD61_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
17B4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD61_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
17B6	TCD Signed Destination Address Offset (eDMA_TCD61_DOFF)	16	R/W	See section	24.3.33/ 1038
17B8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD61_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
17BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD61_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
17BC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD61_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
17BE	TCD Control and Status (eDMA_TCD61_CSR)	16	R/W	See section	24.3.37/ 1041
17C0	TCD Source Address (eDMA_TCD62_SADDR)	32	R/W	See section	24.3.23/ 1030
17C4	TCD Transfer Attributes (eDMA_TCD62_ATTR)	16	R/W	See section	24.3.24/ 1030

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17C6	TCD Signed Source Address Offset (eDMA_TCD62_SOFF)	16	R/W	See section	24.3.25/ 1031
17C8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD62_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
17C8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD62_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
17C8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD62_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
17CC	TCD Last Source Address Adjustment (eDMA_TCD62_SLAST)	32	R/W	See section	24.3.29/ 1035
17D0	TCD Destination Address (eDMA_TCD62_DADDR)	32	R/W	See section	24.3.30/ 1035
17D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD62_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036
17D4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD62_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
17D6	TCD Signed Destination Address Offset (eDMA_TCD62_DOFF)	16	R/W	See section	24.3.33/ 1038
17D8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD62_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
17DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD62_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
17DC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD62_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
17DE	TCD Control and Status (eDMA_TCD62_CSR)	16	R/W	See section	24.3.37/ 1041
17E0	TCD Source Address (eDMA_TCD63_SADDR)	32	R/W	See section	24.3.23/ 1030
17E4	TCD Transfer Attributes (eDMA_TCD63_ATTR)	16	R/W	See section	24.3.24/ 1030
17E6	TCD Signed Source Address Offset (eDMA_TCD63_SOFF)	16	R/W	See section	24.3.25/ 1031
17E8	TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD63_NBYTES_MLNO)	32	R/W	See section	24.3.26/ 1032
17E8	TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD63_NBYTES_MLOFFNO)	32	R/W	See section	24.3.27/ 1032
17E8	TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD63_NBYTES_MLOFFYES)	32	R/W	See section	24.3.28/ 1033
17EC	TCD Last Source Address Adjustment (eDMA_TCD63_SLAST)	32	R/W	See section	24.3.29/ 1035
17F0	TCD Destination Address (eDMA_TCD63_DADDR)	32	R/W	See section	24.3.30/ 1035
17F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD63_CITER_ELINKYES)	16	R/W	See section	24.3.31/ 1036

Table continues on the next page...

eDMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17F4	TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD63_CITER_ELINKNO)	16	R/W	See section	24.3.32/ 1037
17F6	TCD Signed Destination Address Offset (eDMA_TCD63_DOFF)	16	R/W	See section	24.3.33/ 1038
17F8	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD63_DLASTSGA)	32	R/W	See section	24.3.34/ 1038
17FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD63_BITER_ELINKYES)	16	R/W	See section	24.3.35/ 1039
17FC	TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD63_BITER_ELINKNO)	16	R/W	See section	24.3.36/ 1040
17FE	TCD Control and Status (eDMA_TCD63_CSR)	16	R/W	See section	24.3.37/ 1041

24.3.1 Control Register (eDMA_CR)

The CR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in four groups (0, 1, 2, 3) of 16 channels each. Group 3 contains channels 63-48; group 2 contains channels 47-32; group 1 contains channels 31-16; and group 0 contains channels 15-0.

Arbitration within a group can be configured to use a fixed-priority or a round-robin scheme. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities (see the DCHPRI*n* registers). In Round-Robin Arbitration mode, the channel priorities are ignored, and channels within each group are cycled through without regard to priority.

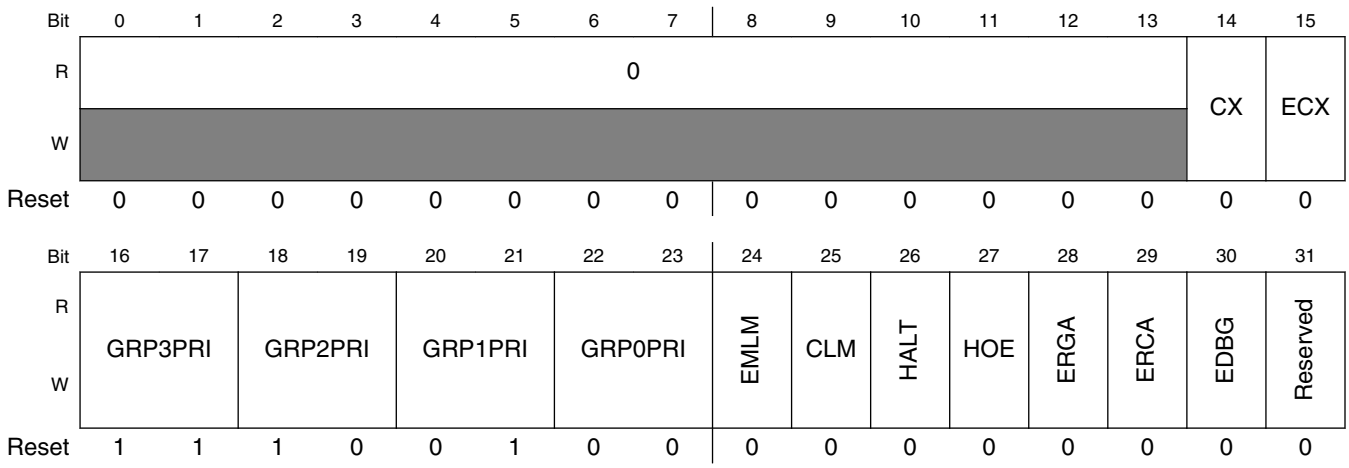
NOTE

Writes to the DMA_CR must be performed only when the DMA channels are inactive (TCDn_CSR[ACTIVE] bits are cleared).

The group priorities operate in a similar fashion. In Group Fixed-Priority Arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to the occurrence of any channel service requests; otherwise, a configuration error will be reported. In Group Round-Robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Memory map/register definition

Address: FC0A_0000h base + 0h offset = FC0A_0000h



eDMA_CR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CX	Cancel transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error cancel transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition; thus updating the ES register and generating an optional error interrupt.
16–17 GRP3PRI	Channel group 3 priority Group 3 priority level when fixed-priority group arbitration is enabled.
18–19 GRP2PRI	Channel group 2 priority Group 2 priority level when fixed-priority group arbitration is enabled.
20–21 GRP1PRI	Channel group 1 priority Group 1 priority level when fixed-priority group arbitration is enabled.
22–23 GRP0PRI	Channel group 0 priority Group 0 priority level when fixed-priority group arbitration is enabled.
24 EMLM	Enable minor loop mapping 0 Disabled. TCD*n*.word2 is defined as a 32-bit NBYTES field 1 Enabled. TCD*n*.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.

Table continues on the next page...

eDMA_CR field descriptions (continued)

Field	Description
25 CLM	Continuous Link mode 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt on error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable round-robin group arbitration 0 Fixed-priority arbitration is used for selection among the groups. 1 Round-robin arbitration is used for selection among the groups.
29 ERCA	Enable round-robin channel arbitration 0 Fixed-priority arbitration is used for channel selection. 1 Round-robin arbitration is used for channel selection.
30 EDBG	Enable debug 0 When in Debug mode the DMA continues to operate. 1 When in Debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system or the EDBG bit is cleared.
31 Reserved	This field is reserved.

24.3.2 Error Status (eDMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode) or an error termination to a bus master read or write cycle.

See Error reporting and handling for more details.

Address: FC0A_0000h base + 4h offset = FC0A_0004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0													UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_ES field descriptions

Field	Description
0 VLD	ERRH and ERRL status bits 0 No ERR bits are set 1 At least one ERR bit is set indicating a valid error exists that has not been cleared
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 UCE	Uncorrectable ECC error 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer cancelled 0 No cancelled transfers 1 The last recorded entry was a cancelled transfer by the error cancel transfer input
16 GPE	Group priority error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique
17 CPE	Channel priority error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities. Channel priorities are not unique.
18–23 ERRCHN	Error channel number or cancelled channel number GPE and/or last recorded error cancelled transfer
24 SAE	Source address error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*_SADDR field. TCD*n*_SADDR is inconsistent with TCD*n*_ATTR[SSIZE]
25 SOE	Source offset error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_SOFF field. TCD*n*_SOFF is inconsistent with TCD*n*_ATTR[SSIZE].

Table continues on the next page...

eDMA_ES field descriptions (continued)

Field	Description
26 DAE	Destination address error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DADDR field. TCD*n*_DADDR is inconsistent with TCD*n*_ATTR[DSIZE].
27 DOE	Destination offset error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DOFF field. TCD*n*_DOFF is inconsistent with TCD*n*_ATTR[DSIZE].
28 NCE	NBYTES/CITER configuration error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_NBYTES or TCD*n*_CITER fields.
29 SGE	Scatter/gather configuration error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD*n*_CSR[ESG] is enabled. TCD*n*_DLASTSGA is not on a 32-byte boundary.
30 SBE	Source bus error 0 No source bus error 1 The last recorded error was a bus error on a source read
31 DBE	Destination bus error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

24.3.3 Enable Request Register High (eDMA_ERQH)

The ERQ{H,L} registers provide a bitmap for the 64 implemented channels to enable the request signal for each channel. ERQH supports channels 63-32, while EQRL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: FC0A_0000h base + 8h offset = FC0A_0008h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ63	ERQ62	ERQ61	ERQ60	ERQ59	ERQ58	ERQ57	ERQ56	ERQ55	ERQ54	ERQ53	ERQ52	ERQ51	ERQ50	ERQ49	ERQ48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ47	ERQ46	ERQ45	ERQ44	ERQ43	ERQ42	ERQ41	ERQ40	ERQ39	ERQ38	ERQ37	ERQ36	ERQ35	ERQ34	ERQ33	ERQ32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_ERQH field descriptions

Field	Description
0 ERQ63	Enable DMA Request 63 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ62	Enable DMA Request 62 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ61	Enable DMA Request 61 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ60	Enable DMA Request 60 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ59	Enable DMA Request 59 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ58	Enable DMA Request 58 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ57	Enable DMA Request 57 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ56	Enable DMA Request 56 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

eDMA_ERQH field descriptions (continued)

Field	Description
8 ERQ55	Enable DMA Request 55 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ54	Enable DMA Request 54 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ53	Enable DMA Request 53 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ52	Enable DMA Request 52 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ51	Enable DMA Request 51 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ50	Enable DMA Request 50 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ49	Enable DMA Request 49 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ48	Enable DMA Request 48 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ47	Enable DMA Request 47 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ46	Enable DMA Request 46 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ45	Enable DMA Request 45 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ44	Enable DMA Request 44 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

eDMA_ERQH field descriptions (continued)

Field	Description
20 ERQ43	Enable DMA Request 43 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ42	Enable DMA Request 42 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ41	Enable DMA Request 41 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ40	Enable DMA Request 40 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ39	Enable DMA Request 39 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ38	Enable DMA Request 38 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ37	Enable DMA Request 37 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ36	Enable DMA Request 36 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ35	Enable DMA Request 35 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ34	Enable DMA Request 34 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ33	Enable DMA Request 33 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ32	Enable DMA Request 32 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

24.3.4 Enable Request Register Low (eDMA_ERQL)

The ERQ{H,L} registers provide a bit map for the 64 implemented channels to enable the request signal for each channel. ERQH supports channels 63-32, while EQRL covers channels 31-00.

The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: FC0A_0000h base + Ch offset = FC0A_000Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_ERQL field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

eDMA_ERQL field descriptions (continued)

Field	Description
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

eDMA_ERQL field descriptions (continued)

Field	Description
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

eDMA_ERQL field descriptions (continued)

Field	Description
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

24.3.5 Enable Error Interrupt Register High (eDMA_EEIH)

The EEI{H,L} registers provide a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63-32, while EEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: FC0A_0000h base + 10h offset = FC0A_0010h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI63	EEI62	EEI61	EEI60	EEI59	EEI58	EEI57	EEI56	EEI55	EEI54	EEI53	EEI52	EEI51	EEI50	EEI49	EEI48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI47	EEI46	EEI45	EEI44	EEI43	EEI42	EEI41	EEI40	EEI39	EEI38	EEI37	EEI36	EEI35	EEI34	EEI33	EEI32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_EEIH field descriptions

Field	Description
0 EEI63	Enable error interrupt 63 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI62	Enable error interrupt 62 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI61	Enable error interrupt 61 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI60	Enable error interrupt 60 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI59	Enable error interrupt 59 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI58	Enable error interrupt 58 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI57	Enable error interrupt 57 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI56	Enable error interrupt 56 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI55	Enable error interrupt 55 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI54	Enable error interrupt 54 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

eDMA_EEIH field descriptions (continued)

Field	Description
10 EEI53	Enable error interrupt 53 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI52	Enable error interrupt 52 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI51	Enable error interrupt 51 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI50	Enable error interrupt 50 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI49	Enable error interrupt 49 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI48	Enable error interrupt 48 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI47	Enable error interrupt 47 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI46	Enable error interrupt 46 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI45	Enable error interrupt 45 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI44	Enable error interrupt 44 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI43	Enable error interrupt 43 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI42	Enable error interrupt 42 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

eDMA_EEIH field descriptions (continued)

Field	Description
22 EEI41	Enable error interrupt 41 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI40	Enable error interrupt 40 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI39	Enable error interrupt 39 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI38	Enable error interrupt 38 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI37	Enable error interrupt 37 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI36	Enable error interrupt 36 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI35	Enable error interrupt 35 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI34	Enable error interrupt 34 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI33	Enable error interrupt 33 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI32	Enable error interrupt 32 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

24.3.6 Enable Error Interrupt Register Low (eDMA_EEIL)

The Enable Error Interrupt Register Low (DMA_EEIL) provides a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63-32, while EEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: FC0A_0000h base + 14h offset = FC0A_0014h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_EEIL field descriptions

Field	Description
0 EEI31	Enable error interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable error interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable error interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable error interrupt 28

Table continues on the next page...

eDMA_EEIL field descriptions (continued)

Field	Description
	0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable error interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable error interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable error interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable error interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI23	Enable error interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable error interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable error interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable error interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable error interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable error interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable error interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable error interrupt 16

Table continues on the next page...

eDMA_EEIL field descriptions (continued)

Field	Description
	0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable error interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable error interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable error interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable error interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable error interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable error interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable error interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable error interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable error interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable error interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable error interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable error interrupt 4

Table continues on the next page...

eDMA_EEIL field descriptions (continued)

Field	Description
	0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable error interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable error interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable error interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable error interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

24.3.7 Set Enable Request Register (eDMA_SERQ)

The Set Enable Request Register (DMA_SERQ) provides a simple memory-mapped mechanism to set a given bit in the ERQ{H,L} to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ{H,L} to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 18h offset = FC0A_0018h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	SAER	SERQ					
Reset	0	0	0	0	0	0	0	0

eDMA_SERQ field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register

Table continues on the next page...

eDMA_SERQ field descriptions (continued)

Field	Description
1 SAER	Set all enable requests 0 Set only those ERQ bits specified in the SERQ field 1 Set all bits in ERQL{H,L}
2–7 SERQ	Set enable request Sets the corresponding bit in ERQL{H,L}

24.3.8 Clear Enable Request Register (eDMA_CERQ)

The Clear Enable Request Register (DMA_CERQ) provides a simple memory-mapped mechanism to clear a given bit in the ERQ{H,L} to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ{H,L} to be cleared, disabling all DMA request inputs.

If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 19h offset = FC0A_0019h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAER	CERQ					
Reset	0	0	0	0	0	0	0	0

eDMA_CERQ field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register
1 CAER	Clear all enable requests 0 Clear only those ERQ bits specified in the CERQ field 1 Clear all bits in ERQ
2–7 CERQ	Clear enable request Clears the corresponding bit in ERQL{H,L}

24.3.9 Set Enable Error Interrupt Register (eDMA_SEEI)

The Set Enable Error Interrupt Register (DMA_SEEI) provides a simple memory-mapped mechanism to set a given bit in the EEI{H,L} to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI{H,L} contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 1Ah offset = FC0A_001Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	SAEE	SEEI					
Reset	0	0	0	0	0	0	0	0

eDMA_SEEI field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register
1 SAEE	Sets all enable error interrupts 0 Set only those EEI bits specified in the SEEI field. 1 Sets all bits in EEI
2–7 SEEI	Set enable error interrupt Sets the corresponding bit in EEIL{H,L}

24.3.10 Clear Enable Error Interrupt Register (eDMA_CEEI)

The Clear Enable Error Interrupt Register (DMA_CEEI) provides a simple memory-mapped mechanism to clear a given bit in the EEI{H,L} to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI{H,L} contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 1Bh offset = FC0A_001Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAEE	CEEI					
Reset	0	0	0	0	0	0	0	0

eDMA_CEEI field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register
1 CAEE	Clear all enable error interrupts 0 Clear only those EEI bits specified in the CEEI field 1 Clear all bits in EEI
2–7 CEEI	Clear enable error interrupt Clears the corresponding bit in EEIL{H,L}

24.3.11 Clear Interrupt Request Register (eDMA_CINT)

The Clear Interrupt Request Register (DMA_CINT) provides a simple, memory-mapped mechanism to clear a given bit in the INT{H,L} to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT{H,L} to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT{H,L} to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 1Ch offset = FC0A_001Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAIR	CINT					
Reset	0	0	0	0	0	0	0	0

eDMA_CINT field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register

Table continues on the next page...

eDMA_CINT field descriptions (continued)

Field	Description
1 CAIR	Clear all interrupt requests 0 Clear only those INT bits specified in the CINT field 1 Clear all bits in INTL{H,L}
2–7 CINT	Clear interrupt request Clears the corresponding bit in INTL{H,L}

24.3.12 Clear Error Register (eDMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR{H,L} to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR{H,L} to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR{H,L} contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 1Dh offset = FC0A_001Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAEI	CERR					
Reset	0	0	0	0	0	0	0	0

eDMA_CERR field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register
1 CAEI	Clear all error indicators 0 Clear only those ERR bits specified in the CERR field 1 Clear all bits in ERR
2–7 CERR	Clear error indicator Clears the corresponding bit in ERRL{H,L}

24.3.13 Set START Bit Register (eDMA_SSRT)

The Set START Bit Register (DMA_SSRT) provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 1Eh offset = FC0A_001Eh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	SAST	SSRT					
Reset	0	0	0	0	0	0	0	0

eDMA_SSRT field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register
1 SAST	Set all START bits (activates all channels) 0 Set only those TCD*n*_CSR[START] bits specified in the SSRT field 1 Set all bits in TCD*n*_CSR[START]
2–7 SSRT	Set START bit Sets the corresponding bit in TCD*n*_CSR[START]

24.3.14 Clear DONE Status Bit Register (eDMA_CDNE)

The Clear DONE Status Bit Register (DMA_CDNE) provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: FC0A_0000h base + 1Fh offset = FC0A_001Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CADN	CDNE					
Reset	0	0	0	0	0	0	0	0

eDMA_CDNE field descriptions

Field	Description
0 NOP	No operation 0 Normal operation 1 No operation, ignore bits 1-7 of this register
1 CADN	Clears all DONE bits 0 Clears only those TCD*n*_CSR[DONE] bits specified in the CDNE field 1 Clears all bits in TCD*n*_CSR[DONE]
2–7 CDNE	Clear DONE bit Clears the corresponding bit in TCD*n*_CSR[DONE]

24.3.15 Interrupt Request Register High (eDMA_INTH)

The Interrupt Request Register High (DMA_INTH) provides a bit map for the upper half of 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptions, the eDMA engine generates an interrupt on a data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L}.

Address: FC0A_0000h base + 20h offset = FC0A_0020h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT39	INT38	INT37	INT36	INT35	INT34	INT33	INT32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_INTH field descriptions

Field	Description
0 INT63	Interrupt request 63 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT62	Interrupt request 62 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT61	Interrupt request 61 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT60	Interrupt request 60 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT59	Interrupt request 59

Table continues on the next page...

eDMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT58	Interrupt request 58 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT57	Interrupt request 57 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT56	Interrupt request 56 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT55	Interrupt request 55 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT54	Interrupt request 54 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT53	Interrupt request 53 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT52	Interrupt request 52 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT51	Interrupt request 51 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT50	Interrupt request 50 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT49	Interrupt request 49 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT48	Interrupt request 48 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT47	Interrupt request 47

Table continues on the next page...

eDMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT46	Interrupt request 46 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT45	Interrupt request 45 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT44	Interrupt request 44 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT43	Interrupt request 43 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT42	Interrupt request 42 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT41	Interrupt request 41 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT40	Interrupt request 40 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT39	Interrupt request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT38	Interrupt request 38 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT37	Interrupt request 37 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT36	Interrupt request 36 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT35	Interrupt request 35

Table continues on the next page...

eDMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT34	Interrupt request 34 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT33	Interrupt request 33 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT32	Interrupt request 32 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

24.3.16 Interrupt Request Register Low (eDMA_INTL)

The Interrupt Request Register Low (DMA_INTL) provides a bit map for the lower half of the 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt at data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L}.

Address: FC0A_0000h base + 24h offset = FC0A_0024h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_INTL field descriptions

Field	Description
0 INT31	Interrupt request 31 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt request 25 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT22	Interrupt request 22

Table continues on the next page...

eDMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt request 19 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt request 13 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt request 10

Table continues on the next page...

eDMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT8	Interrupt request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt request 7 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

24.3.17 Error Register High (eDMA_ERRH)

The ERR{H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63-32, while ERRL covers channels 31-00. The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a nonzero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: FC0A_0000h base + 28h offset = FC0A_0028h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR63	ERR62	ERR61	ERR60	ERR59	ERR58	ERR57	ERR56	ERR55	ERR54	ERR53	ERR52	ERR51	ERR50	ERR49	ERR48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR47	ERR46	ERR45	ERR44	ERR43	ERR42	ERR41	ERR40	ERR39	ERR38	ERR37	ERR36	ERR35	ERR34	ERR33	ERR32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_ERRH field descriptions

Field	Description
0 ERR63	Error in channel 63 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
1 ERR62	Error in channel 62 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
2 ERR61	Error in channel 61 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
3 ERR60	Error in channel 60 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
4 ERR59	Error in channel 59 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
5 ERR58	Error in channel 58 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
6 ERR57	Error in channel 57 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
7 ERR56	Error in channel 56 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
8 ERR55	Error in channel 55 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
9 ERR54	Error in channel 54 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
10 ERR53	Error in channel 53 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
11 ERR52	Error in channel 52 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table continues on the next page...

eDMA_ERRH field descriptions (continued)

Field	Description
12 ERR51	Error in channel 51 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
13 ERR50	Error in channel 50 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
14 ERR49	Error in channel 49 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
15 ERR48	Error in channel 48 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
16 ERR47	Error in channel 47 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
17 ERR46	Error in channel 46 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
18 ERR45	Error in channel 45 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
19 ERR44	Error in channel 44 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
20 ERR43	Error in channel 43 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
21 ERR42	Error in channel 42 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
22 ERR41	Error in channel 41 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
23 ERR40	Error in channel 40 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table continues on the next page...

eDMA_ERRH field descriptions (continued)

Field	Description
24 ERR39	Error in channel 39 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
25 ERR38	Error in channel 38 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
26 ERR37	Error in channel 37 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
27 ERR36	Error in channel 36 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
28 ERR35	Error in channel 35 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
29 ERR34	Error in channel 34 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
30 ERR33	Error in channel 33 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
31 ERR32	Error in channel 32 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

24.3.18 Error Register Low (eDMA_ERRL)

The ERR{H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63-32, while ERRL covers channels 31-00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: FC0A_0000h base + 2Ch offset = FC0A_002Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_ERRL field descriptions

Field	Description
0 ERR31	Error in channel 31 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
1 ERR30	Error in channel 30 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
2 ERR29	Error in channel 29 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
3 ERR28	Error in channel 28 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
4 ERR27	Error in channel 27 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
5 ERR26	Error in channel 26 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
6 ERR25	Error in channel 25 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
7 ERR24	Error in channel 24 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
8 ERR23	Error in channel 23 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
9 ERR22	Error in channel 22 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
10 ERR21	Error in channel 21 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
11 ERR20	Error in channel 20 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table continues on the next page...

eDMA_ERRL field descriptions (continued)

Field	Description
12 ERR19	Error in channel 19 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
13 ERR18	Error in channel 18 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
14 ERR17	Error in channel 17 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
15 ERR16	Error in channel 16 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
16 ERR15	Error in channel 15 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
17 ERR14	Error in channel 14 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
18 ERR13	Error in channel 13 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
19 ERR12	Error in channel 12 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
20 ERR11	Error in channel 11 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
21 ERR10	Error in channel 10 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
22 ERR9	Error in channel 9 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
23 ERR8	Error in channel 8 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table continues on the next page...

eDMA_ERRL field descriptions (continued)

Field	Description
24 ERR7	Error in channel 7 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
25 ERR6	Error in channel 6 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
26 ERR5	Error in channel 5 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
27 ERR4	Error in channel 4 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
28 ERR3	Error in channel 3 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
29 ERR2	Error in channel 2 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
30 ERR1	Error in channel 1 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
31 ERR0	Error in channel 0 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

24.3.19 Hardware Request Status Register High (eDMA_HRSH)

The HRS{H,L} provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSH supports channels 63-32, while HRSL covers channels 31-00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQ{H,L} fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.

Address: FC0A_0000h base + 30h offset = FC0A_0030h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HRS63	HRS62	HRS61	HRS60	HRS59	HRS58	HRS57	HRS56	HRS55	HRS54	HRS53	HRS52	HRS51	HRS50	HRS49	HRS48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS47	HRS46	HRS45	HRS44	HRS43	HRS42	HRS41	HRS40	HRS39	HRS38	HRS37	HRS36	HRS35	HRS34	HRS33	HRS32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_HRSH field descriptions

Field	Description
0 HRS63	Hardware request status channel 63 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS62	Hardware request status channel 62 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS61	Hardware request status channel 61 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS60	Hardware request status channel 60 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS59	Hardware request status channel 59 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS58	Hardware request status channel 58 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS57	Hardware request status channel 57

Table continues on the next page...

eDMA_HRSH field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS56	Hardware request status channel 56 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS55	Hardware request status channel 55 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS54	Hardware request status channel 54 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS53	Hardware request status channel 53 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS52	Hardware request status channel 52 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS51	Hardware request status channel 51 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS50	Hardware request status channel 50 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS49	Hardware request status channel 49 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS48	Hardware request status channel 48 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS47	Hardware request status channel 47 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS46	Hardware request status channel 46 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS45	Hardware request status channel 45

Table continues on the next page...

eDMA_HRSH field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS44	Hardware request status channel 44 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS43	Hardware request status channel 43 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS42	Hardware request status channel 42 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS41	Hardware request status channel 41 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS40	Hardware request status channel 40 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS39	Hardware request status channel 39 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS38	Hardware request status channel 38 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS37	Hardware request status channel 37 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS36	Hardware request status channel 36 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS35	Hardware request status channel 35 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS34	Hardware request status channel 34 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS33	Hardware request status channel 33

Table continues on the next page...

eDMA_HRSH field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS32	Hardware request status channel 32 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

24.3.20 Hardware Request Status Register Low (eDMA_HRSL)

The HRS{H,L} provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSH supports channels 63-32, while HRSL covers channels 31-00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQ{H,L} fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.

Address: FC0A_0000h base + 34h offset = FC0A_0034h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eDMA_HRSL field descriptions

Field	Description
0 HRS31	Hardware request status channel 31

Table continues on the next page...

eDMA_HRSL field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS30	Hardware request status channel 30 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS29	Hardware request status channel 29 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS28	Hardware request status channel 28 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS27	Hardware request status channel 27 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS26	Hardware request status channel 26 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS25	Hardware request status channel 25 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS24	Hardware request status channel 24 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS23	Hardware request status channel 23 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS22	Hardware request status channel 22 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS21	Hardware request status channel 21 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS20	Hardware request status channel 20 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS19	Hardware request status channel 19

Table continues on the next page...

eDMA_HRSL field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS18	Hardware request status channel 18 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS17	Hardware request status channel 17 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS16	Hardware request status channel 16 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS15	Hardware request status channel 15 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS14	Hardware request status channel 14 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS13	Hardware request status channel 13 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS12	Hardware request status channel 12 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS11	Hardware request status channel 11 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS10	Hardware request status channel 10 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS9	Hardware request status channel 9 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS8	Hardware request status channel 8 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS7	Hardware request status channel 7

Table continues on the next page...

eDMA_HRSL field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS6	Hardware request status channel 6 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS5	Hardware request status channel 5 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS4	Hardware request status channel 4 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS3	Hardware request status channel 3 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS2	Hardware request status channel 2 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS1	Hardware request status channel 1 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS0	Hardware request status channel 0 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

24.3.21 Channel Priority Register (eDMA_DCHPRI*n*)

When the Fixed-Priority Channel Arbitration mode is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values. Otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

When read, the GRPPRI bits of the DCHPRI*n* register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRI*n* registers. The group priority is assigned in the DMA control register.

Address: FC0A_0000h base + 100h offset + (1d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read Write	ECP	DPA	GRPPRI		CHPRI			
Reset	0	0	*	*	0	0	0	0

* Notes:
- GRPPRI field: Undefined at reset

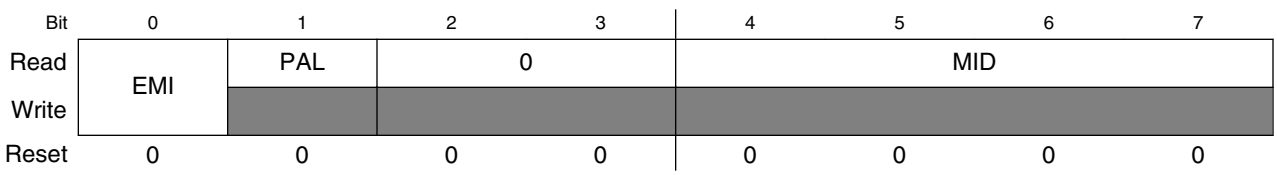
eDMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable channel preemption This bit resets to zero. 0 Channel *n* cannot be suspended by a higher priority channel's service request 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel
1 DPA	Disable preempt ability This bit resets to zero. 0 Channel *n* can suspend a lower priority channel 1 Channel *n* cannot suspend any channel, regardless of channel priority
2–3 GRPPRI	Channel *n* current group priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, i.e., DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4–7 CHPRI	Channel *n* arbitration priority Channel priority when fixed-priority arbitration is enabled **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, i.e., DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.

24.3.22 Channel Master ID Register (eDMA_DCHMID*n*)

The DMA Master ID Replication registers allow the DMA to use the same protection level and AHB system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master (a core, for example) programs a TCD, its master ID and protection level are captured when the TCD Word 7 control attributes are written. Although the scatter/gather operation can change the contents of TCD Word 7, that operation does not affect the DCHMID*n* registers.

Address: FC0A_0000h base + 140h offset + (1d × i), where i=0d to 63d



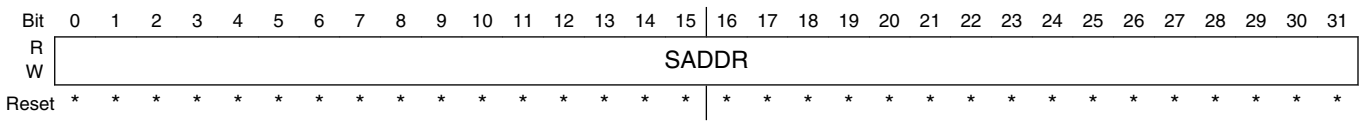
eDMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication 0 Master ID replication is disabled 1 Master ID replication is enabled
1 PAL	Privileged Access Level The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCD Word 7. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MID	Master ID DMA's master ID when channel *n* is active and master ID replication is enabled. **NOTE:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCD Word 7.

24.3.23 TCD Source Address (eDMA_TCD*n*_SADDR)

See TCD Source Address register figure and DMA_TCDn_SADDR field descriptions table as follows.

Address: FC0A_0000h base + 1000h offset + (32d × i), where i=0d to 63d



* Notes:
- SADDR field: Undefined at reset

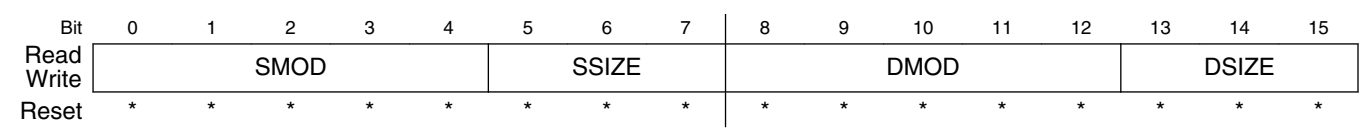
eDMA_TCD*n*_SADDR field descriptions

Field	Description
0–31 SADDR	Source address Memory address pointing to the source data.

24.3.24 TCD Transfer Attributes (eDMA_TCD*n*_ATTR)

See TCD Transfer Attributes register figure and DMA_TCDn_ATTR field descriptions table as follows.

Address: FC0A_0000h base + 1004h offset + (32d × i), where i=0d to 63d



* Notes:
- DSIZE field: Undefined at reset
- DMOD field: Undefined at reset
- SSIZE field: Undefined at reset
- SMOD field: Undefined at reset

eDMA_TCD*n*_ATTR field descriptions

Field	Description
0–4 SMOD	Source address modulo. 0: Source address modulo feature is disabled

Table continues on the next page...

eDMA_TCD*n*_ATTR field descriptions (continued)

Field	Description
	Not 0: This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5–7 SSIZE	Source data transfer size Using a reserved encoding causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 Reserved 100 16-byte 101 Reserved 110 Reserved 111 Reserved
8–12 DMOD	Destination address modulo See the SMOD definition
13–15 DSIZE	Destination data transfer size See the SSIZE definition

24.3.25 TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)

See TCD Signed Source Address Offset register figure and DMA_TCDn_SOFF field descriptions table as follows.

Address: FC0A_0000h base + 1006h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SOFF															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- SOFF field: Undefined at reset

eDMA_TCD*n*_SOFF field descriptions

Field	Description
0–15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

24.3.26 TCD Minor Byte Count Minor Loop Disabled (eDMA_TCD*n*_NBYTES_MLNO)

See TCD Minor Byte Count (Minor Loop Disabled) register figure and DMA_TCDn_NBYTES_MLNO field descriptions table below.

If minor loop mapping is disabled (CR[EMLM] = 0), TCD word 2 is defined as follows.

Address: FC0A_0000h base + 1008h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- NBYTES field: Undefined at reset

eDMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
0–31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

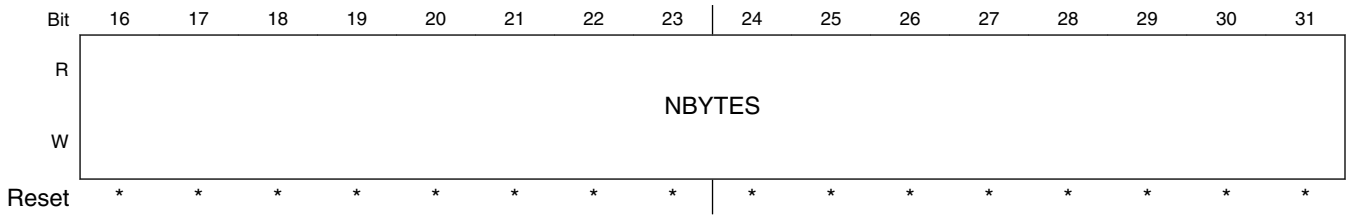
24.3.27 TCD Signed Minor Loop Offset Minor Loop Enabled and Offset Disabled (eDMA_TCD*n*_NBYTES_MLOFFNO)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

Address: FC0A_0000h base + 1008h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMLOE	DMLOE	NBYTES													
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*



* Notes:
- NBYTES field: Undefined at reset
- DMLOE field: Undefined at reset
- SMLOE field: Undefined at reset

eDMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

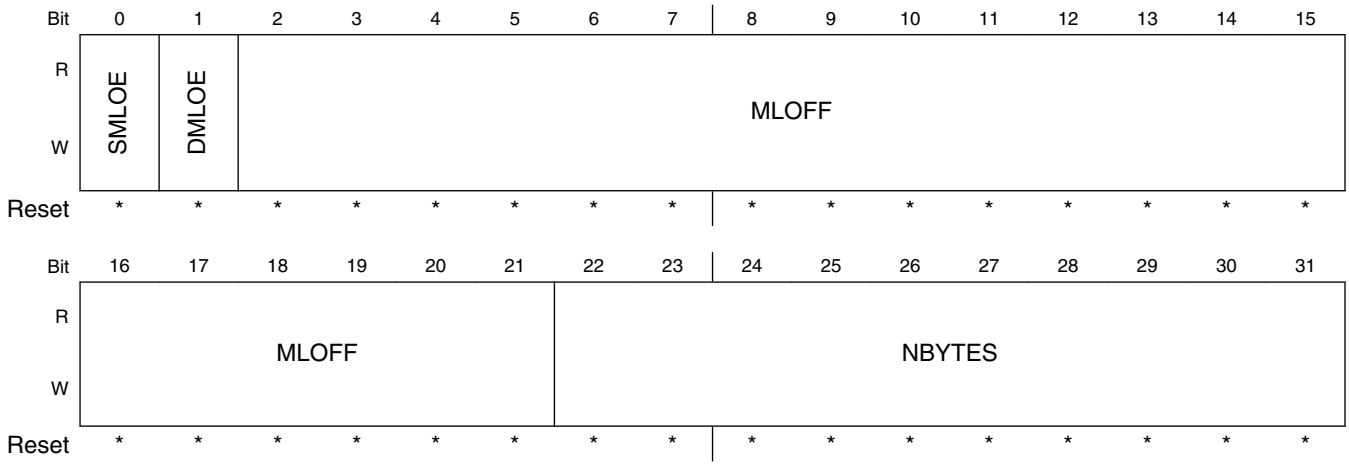
24.3.28 TCD Signed Minor Loop Offset Minor Loop and Offset Enabled (eDMA_TCD*n*_NBYTES_MLOFFYES)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset enabled (SMLOE or DMLOE = 1)

Memory map/register definition

Address: FC0A_0000h base + 1008h offset + (32d × i), where i=0d to 63d



* Notes:
- NBYTES field: Undefined at reset
- MLOFF field: Undefined at reset
- DMLOE field: Undefined at reset
- SMLOE field: Undefined at reset

eDMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22–31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

24.3.29 TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)

Address: FC0A_0000h base + 100Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- SLAST field: Undefined at reset

eDMA_TCD*n*_SLAST field descriptions

Field	Description
0–31 SLAST	Last source address adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.

24.3.30 TCD Destination Address (eDMA_TCD*n*_DADDR)

See TCD Destination Address register figure and DMA_TCDn_DADDR field descriptions table as follows.

Address: FC0A_0000h base + 1010h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- DADDR field: Undefined at reset

eDMA_TCD*n*_DADDR field descriptions

Field	Description
0–31 DADDR	Destination address Memory address pointing to the destination data.

24.3.31 TCD Current Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD*n*_CITER_ELINKYES)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) register figure and DMA_TCDn_CITER_ELINKYES field descriptions table as follows.

Address: FC0A_0000h base + 1014h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	LINKCH						CITER
Reset	*	*	*	*	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read Write	CITER							
Reset	*	*	*	*	*	*	*	*

* Notes:
- CITER field: Undefined at reset
- LINKCH field: Undefined at reset
- ELINK field: Undefined at reset

eDMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.
7–15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field.

Table continues on the next page...

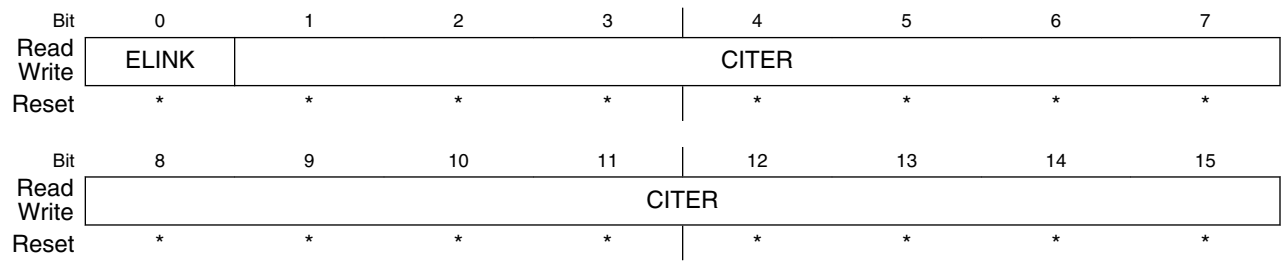
eDMA_TCD*n*_CITER_ELINKYES field descriptions (continued)

Field	Description
	NOTE: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

24.3.32 TCD Current Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD*n*_CITER_ELINKNO)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) register figure and DMA_TCDn_CITER_ELINKNO field descriptions table as follows.

Address: FC0A_0000h base + 1014h offset + (32d × i), where i=0d to 63d



* Notes:
- CITER field: Undefined at reset
- ELINK field: Undefined at reset

eDMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel.If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example, final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field.

Table continues on the next page...

eDMA_TCD*n*_CITER_ELINKNO field descriptions (continued)

Field	Description
	NOTE: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

24.3.33 TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)

See TCD Signed Destination Address Offset register figure and DMA_TCDn_DOFF field descriptions table as follows.

Address: FC0A_0000h base + 1016h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DOFF															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- DOFF field: Undefined at reset

eDMA_TCD*n*_DOFF field descriptions

Field	Description
0–15 DOFF	Destination address signed offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

24.3.34 TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)

Address: FC0A_0000h base + 1018h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- DLASTSGA field: Undefined at reset

eDMA_TCD*n*_DLASTSGA field descriptions

Field	Description
0–31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).

eDMA_TCD*n*_DLASTSGA field descriptions (continued)

Field	Description

	If (TCD*n*_CSR[ESG] = 0) then
	• Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.
	else
	• This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, else a configuration error is reported.

24.3.35 TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Enabled (eDMA_TCD*n*_BITER_ELINKYES)

Address: FC0A_0000h base + 101Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	LINKCH						BITER
Reset	*	*	*	*	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	*	*	*	*	*	*	*	*

* Notes:
- BITER field: Undefined at reset
- LINKCH field: Undefined at reset
- ELINK field: Undefined at reset

eDMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–6 LINKCH	Link channel number

Table continues on the next page...

eDMA_TCD*n*_BITER_ELINKYES field descriptions (continued)

Field	Description
	If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.
7–15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

24.3.36 TCD Beginning Minor Loop Link, Major Loop Count Channel Linking Disabled (eDMA_TCD*n*_BITER_ELINKNO)

Address: FC0A_0000h base + 101Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	BITER						
Reset	*	*	*	*	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	*	*	*	*	*	*	*	*

* Notes:
- BITER field: Undefined at reset
- ELINK field: Undefined at reset

eDMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.

Table continues on the next page...

eDMA_TCD*n*_BITER_ELINKNO field descriptions (continued)

Field	Description
	0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

24.3.37 TCD Control and Status (eDMA_TCD*n*_CSR)

See TCD Control and Status register figure and DMA_TCDn_CSR field descriptions table as follows.

Address: FC0A_0000h base + 101Eh offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read Write	BWC		MAJORLINKCH					
Reset	*	*	*	*	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read Write	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Reset	*	*	*	*	*	*	*	*

* Notes:
- START field: Undefined at reset
- INTMAJOR field: Undefined at reset
- INTHALF field: Undefined at reset
- DREQ field: Undefined at reset
- ESG field: Undefined at reset
- MAJORELINK field: Undefined at reset
- ACTIVE field: Undefined at reset
- DONE field: Undefined at reset
- MAJORLINKCH field: Undefined at reset
- BWC field: Undefined at reset

eDMA_TCD*n*_CSR field descriptions

Field	Description
0–1 BWC	Bandwidth control

Table continues on the next page...

eDMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	Throttles the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch (XBAR). **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each r/w 11 eDMA engine stalls for 8 cycles after each r/w

2–7
MAJORLINKCH
	Link channel number
	If (MAJORELINK = 0) then
	• No channel-to-channel linking (or chaining) is performed after the major loop counter is exhausted.
	else
	• After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.

8 DONE	Channel done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero; The software clears it, or the hardware when the channel is activated. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits. This bit resets to zero.
9 ACTIVE	Channel active This flag signals the channel is currently in execution. It is set when channel service begins, and the eDMA clears it as the minor loop completes or if any error condition is detected.
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
11 ESG	Enable scatter/gather processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.

Table continues on the next page...

eDMA_TCD*n*_CSR field descriptions (continued)

Field	Description
12 DREQ	Disable request {H,L} 0 The channel's ERQ bit is not affected 1 The channel's ERQ bit is cleared when the major loop is complete
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER is set, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled 1 The half-point interrupt is enabled
14 INTMAJOR	Enable an interrupt when major iteration count completes If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled 1 The end-of-major loop interrupt is enabled
15 START	Channel start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. This bit resets to zero. 0 The channel is not explicitly started 1 The channel is explicitly started via a software initiated service request

24.4 Functional description

This section provides a description of the DMA request sources, an overview of the microarchitecture, and functional operation of the eDMA module.

24.4.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules:

- eDMA engine
  - Address path:

This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n* _{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including updates to the final address pointer, reloading the TCD*n* _CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.

- Data path:

  This block implements the bus master read/write datapath. It includes 32 bytes of register storage and the necessary multiplex logic to support any data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.

  The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), while the data path module implements the second stage of the pipeline (data phase).

- Program model/channel arbitration:

  This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus (not shown). The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).

- Control:

This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- Transfer control descriptor memory
  - Memory controller:

    This logic implements the dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus.

    As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
  - Memory array:

    TCD storage is implemented using a single-port, synchronous RAM array.

24.4.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in Figure 24-2, the first segment involves the channel activation. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, channel arbitration is performed, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

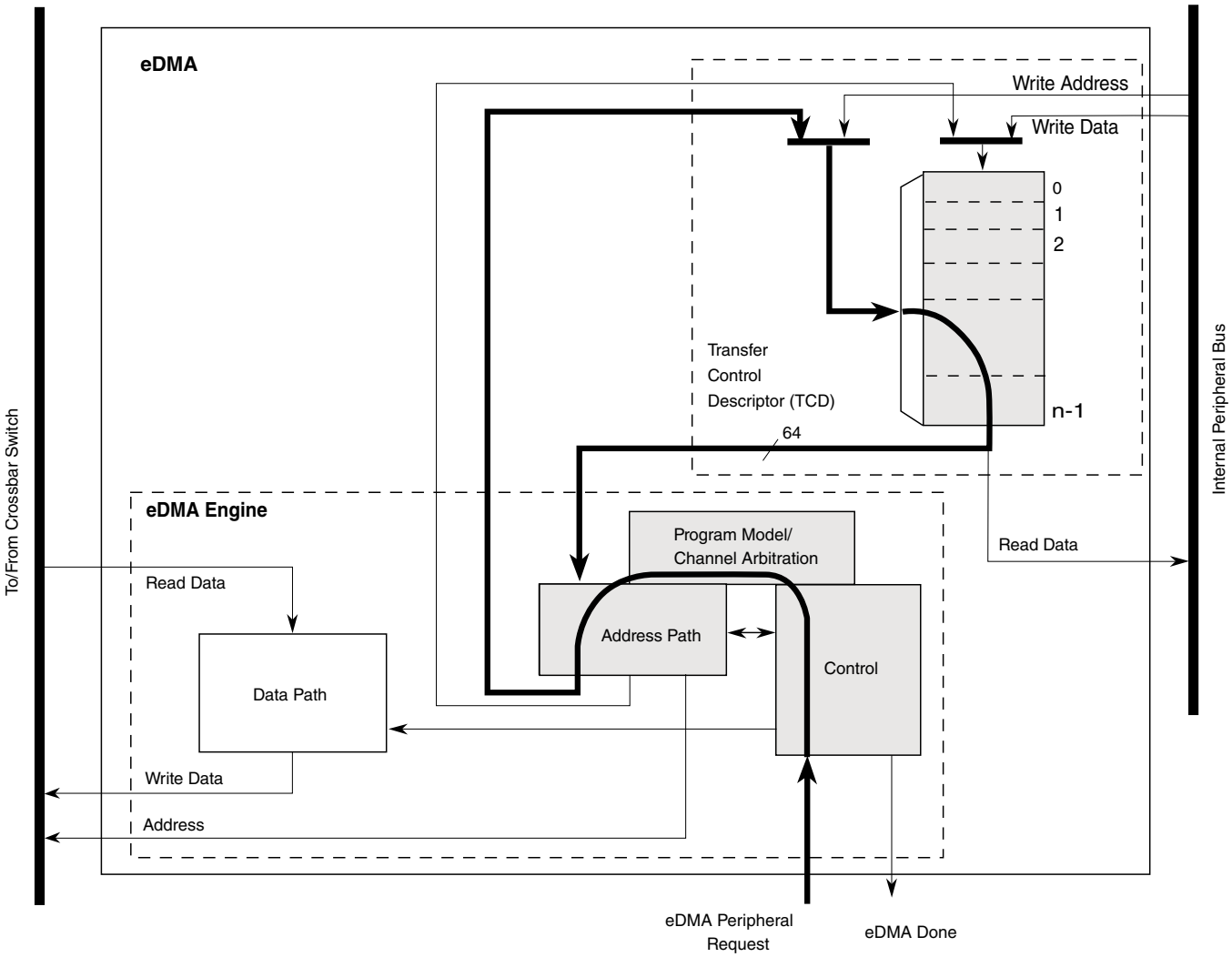


Figure 24-2. DMA operation, part 1

In the second part of the basic data flow as shown in Figure 24-3 the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

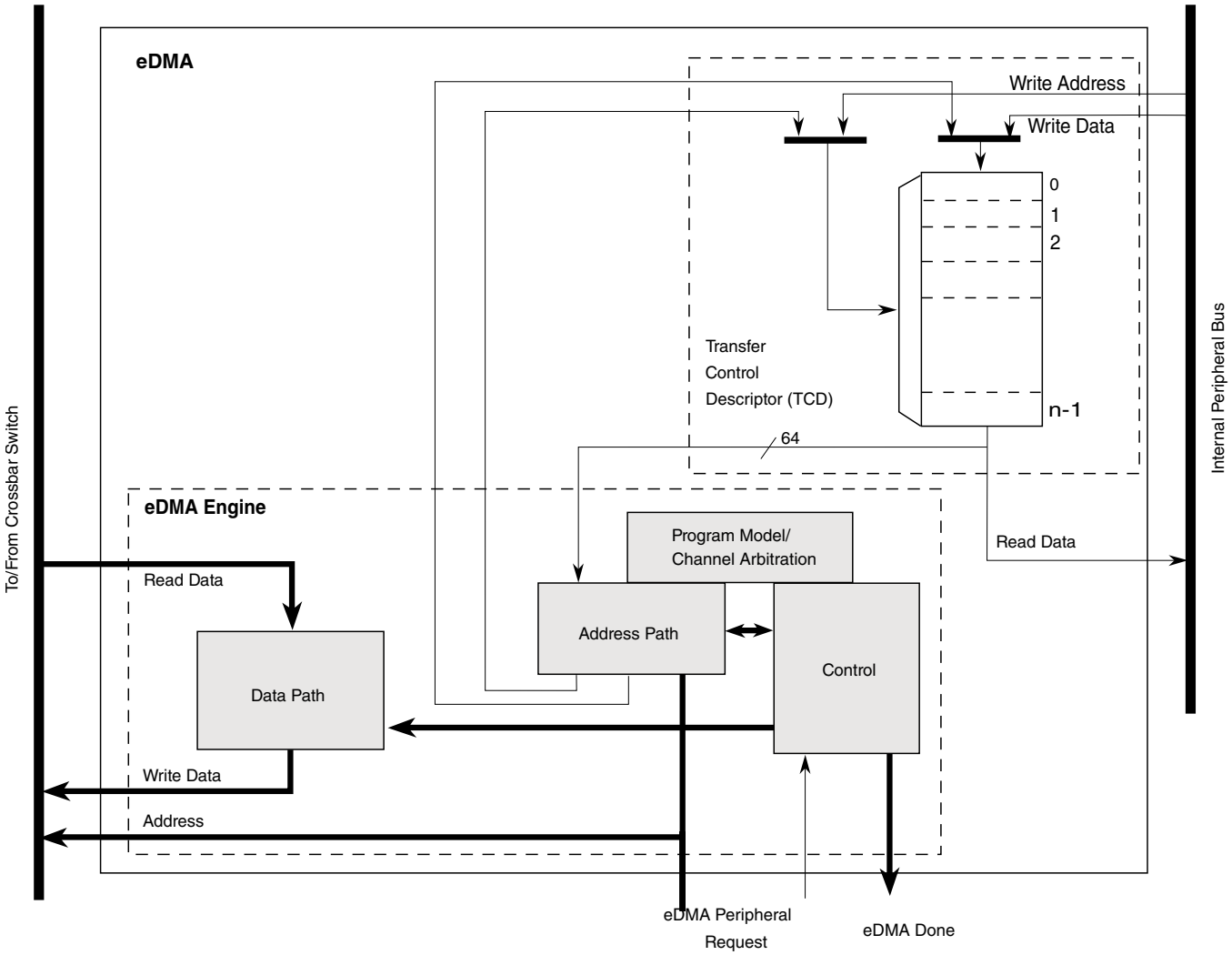


Figure 24-3. DMA operation, part 2

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, and CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in Figure 24-4.

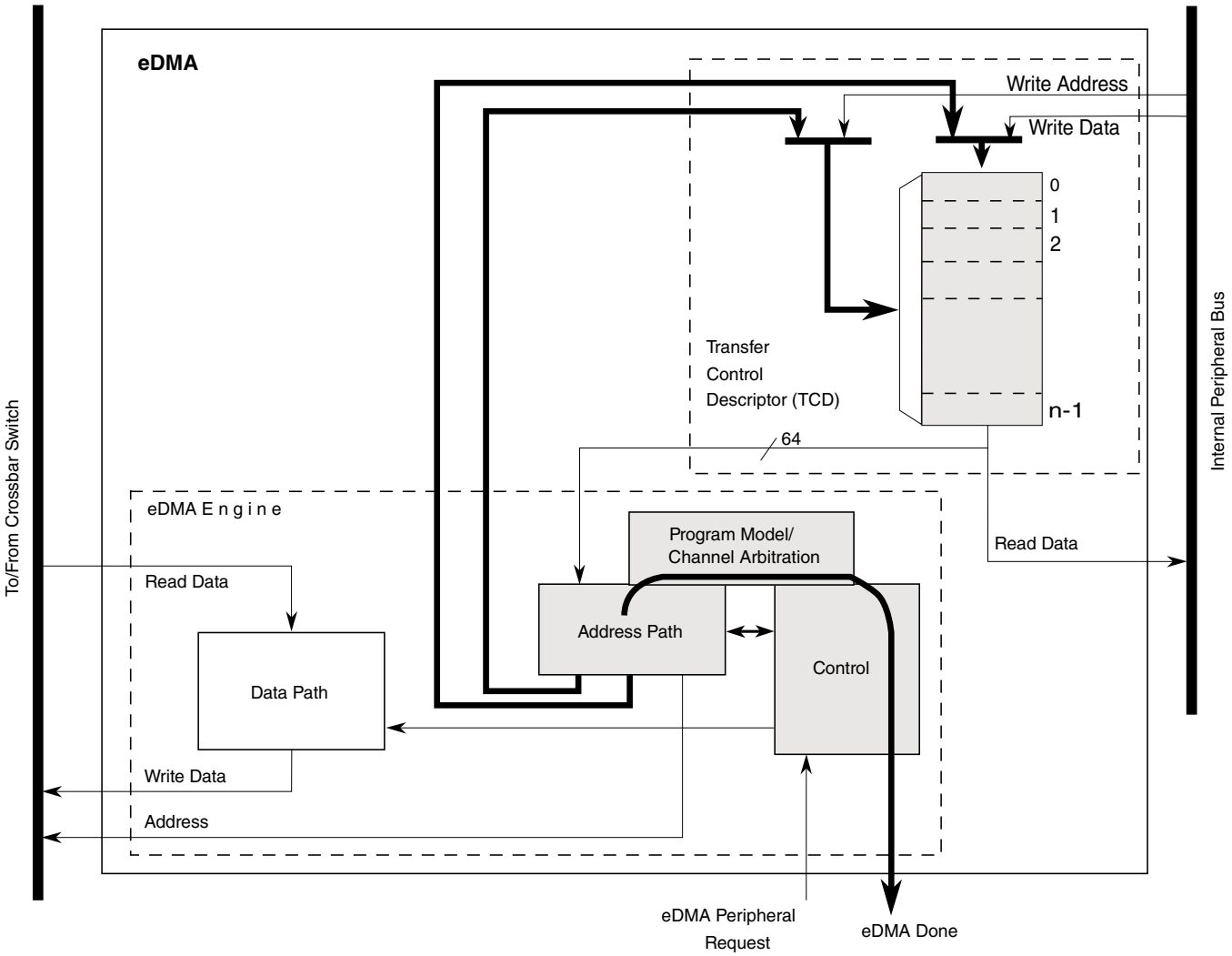


Figure 24-4. DMA operation, part 3

24.4.3 Error reporting and handling

Channel errors are reported in the ES register and can be caused by:

- A configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in fixed-arbitration mode),
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle.

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed in the list below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size, respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled.
- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD*n*_CITER[E_LINK] bit does not equal the TCD*n*_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system-bus error occurs, the channel terminates after the read or write transaction (which is already pipelined after errant access) has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the ES register is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD since the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

The occurrence of any error causes the eDMA engine to stop the active channel immediately, and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the ES register. The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

24.4.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRI*n*[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting DCHPRI*n*[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low-priority, large data-moving channels to be

defined. These low-priority channels can be configured so that they do not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true, high-priority channel.

24.4.5 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in Table 24-1 and Table 24-2. The following assumptions apply to those tables:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait-state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- All internal peripheral bus accesses are 32 bits in size.
- All internal SRAM accesses are 64 bits in size.

Table 24-1 presents a peak transfer rate comparison, measured in megabytes per second.

Table 24-1. eDMA peak transfer rates (MB/s)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
100 MHz, 64 bits	400.0	160.0	160.0
150 MHz, 64 bits	600.0	240.0	240.0
200 MHz, 64 bits	800.0	320.0	320.0

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from

internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The eDMA design supports the following hardware service request sequence:

- Cycle 1: eDMA peripheral request is asserted.
- Cycle 2: The eDMA peripheral request is registered locally in the eDMA module and qualified. (TCD*n* _CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* Word 7).
- Cycle 3: Channel arbitration begins.
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycles 5–6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycles 8 to *x* : The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.

  The exact timing from this point is a function of the transfer size and response times for the channel's read and write accesses. In this case of two internal peripheral bus reads and a single 64-bit internal SRAM write, the combined data phase time is five cycles. For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles.
- Cycle *x*[1] + 1: This cycle represents the data phase of the last destination write.
- Cycle *x*[1] + 2: The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* Word 7 is read and checked for channel linking or scatter/gather requests.
- Cycle *x*[1] + 3: The appropriate fields in the first part of the TCD*n* are written back into the local memory.

40. "x" represents an undefined cycle value dependant on transfer sizes and response times for the channel's read and write accesses.

- Cycle x[1] + 4: The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle *x*[1] + 5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every nine cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (three cycles) and SRAM-to-internal peripheral bus (three cycles), DMA requests can be processed every 10 cycles (4 + (3 + 3)/2 + 3). This is the time from Cycle 4 to Cycle x + 5. The resulting peak request rate, as a function of the system frequency, is shown in Table 24-2. This metric represents millions of requests per second. A single read/write operation with same transfer size for the source and destination is assumed.

Table 24-2. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
100.0	11.1	10.0
150.0	16.6	15.0
200.0	22.2	20.0

A general formula to compute the peak request rate (with overlapping requests) is:

$$PEAKreq = freq/[entry + (1 + read\_ws) + (1 + write\_ws) + exit]$$

where:

- PEAKreq = peak request rate
- freq = system frequency
- entry = channel startup (4 cycles)
- read_ws = wait states seen during the system bus read data phase
- write_ws = wait states seen during the system bus write data phase
- exit = channel shutdown (3 cycles)

For example: consider a system with the following characteristics:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.

- All internal peripheral bus reads require one wait state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- System operates at 150 MHz.

For an SRAM to internal peripheral bus transfer,

$$\text{PEAKreq} = 150\,\text{MHz}/[4+(1+0)+(1+1)+3]\text{cycles} = 15.0\,\text{Mreq/s}$$

For an internal peripheral bus-to-SRAM transfer,

$$\text{PEAKreq} = 150\,\text{MHz}/[4+(1+1)+(1+0)+3]\text{cycles} = 15.0\,\text{Mreq/s}$$

The minimum number of cycles to perform a single read/write, with zero wait states on the system bus, from a cold start (where no channel is executing and the eDMA is idle):

- 11 cycles for a software (TCD*n*_CSR[START] bit) request
- 12 cycles for a hardware (eDMA peripheral request signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering are absorbed in or overlap the previous executing channel.

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

24.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

24.5.1 eDMA initialization

A typical initialization of the eDMA has the following sequence:

1. Write the CR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI registers if so desired.

4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service by software (setting the TCD*n*_CSR[START] bit) or hardware (slave device asserting its eDMA peripheral request signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields (shown in Table 24-3) for the selected channel into its internal address path module. As the TCD is read, the first transfer is initiated on the internal bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD*n*_SADDR) to the destination (as defined by the destination address, TCD*n*_DADDR) continue until the specified number of bytes (TCD*n*_NBYTES) are transferred. When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes (interrupts, major loop channel linking, and scatter/gather operations) if enabled.

Table 24-3. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

Figure 24-5 shows how each DMA request initiates one minor-loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

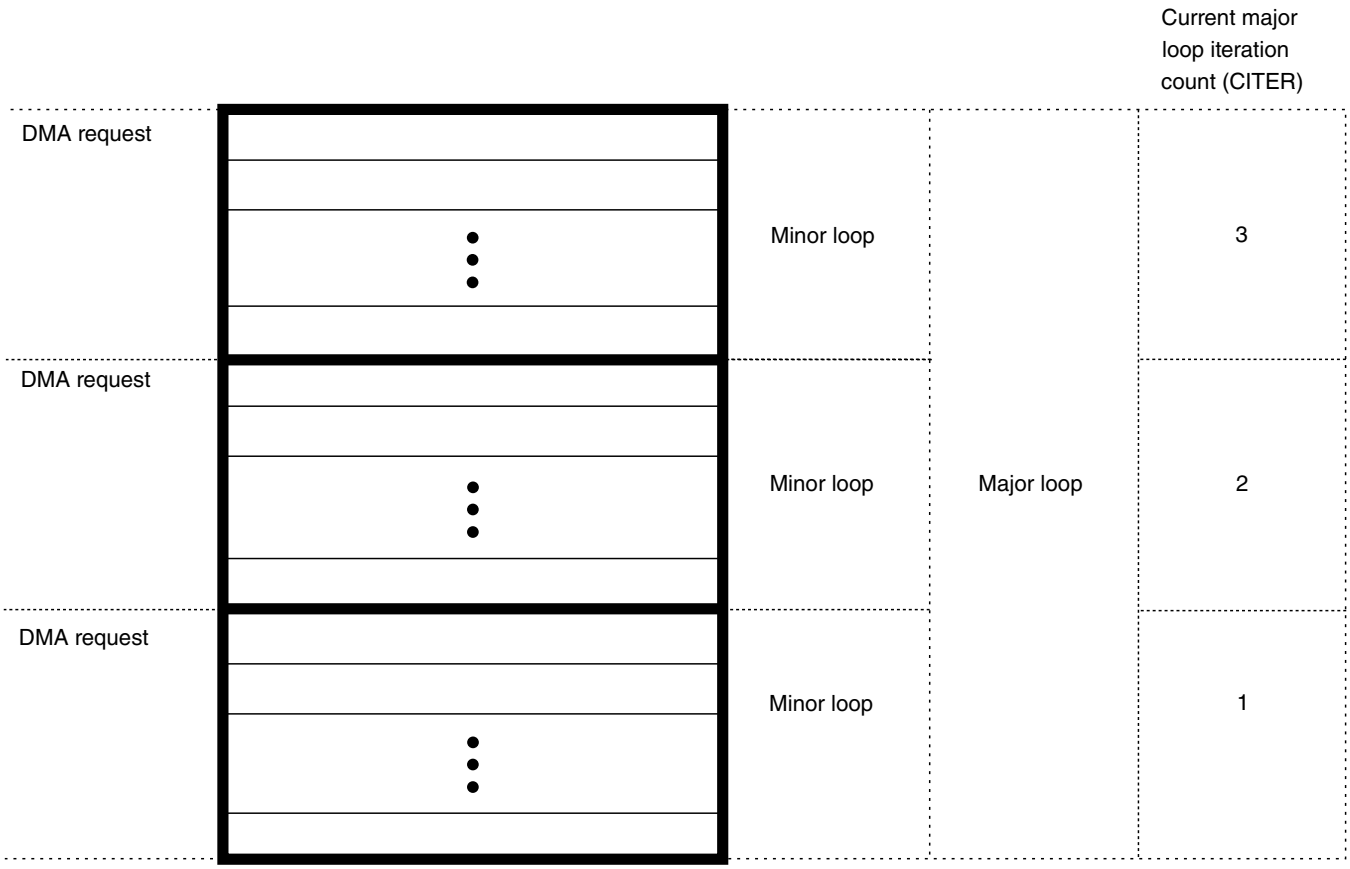


Figure 24-5. Example of multiple loop iterations

Figure 24-6 lists the memory array terms and how the TCD settings interrelate.

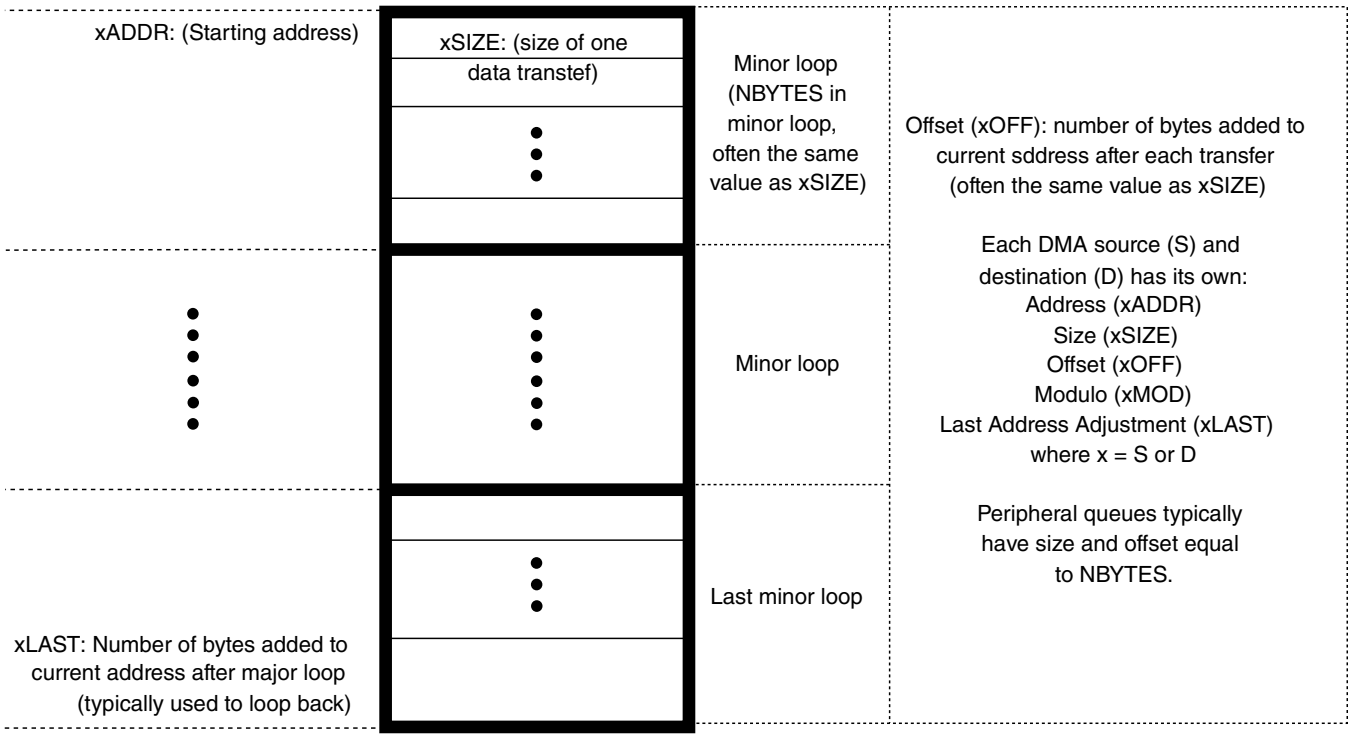


Figure 24-6. Memory array terms

24.5.2 DMA programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of two errors: group priority error (ES[GPE]) and channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the ES register. If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed-group and fixed-channel arbitration modes.
2. Group 3 is the highest priority and all channels are unique in that group.
3. Group 2 is the next highest priority and has two channels with the same priority level.
4. If Group 3 has any service requests, those requests will be executed.

5. Once all of Group 3 requests have completed, Group 02 will be the next active group.
6. If Group 2 has a service request, then an undefined channel in Group 02 will be selected and a channel priority error will occur.
7. This repeats until all of the Group 02 requests have been removed or a higher priority Group 13 request comes in.

In this sequence, for item 2, the eDMA Acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest (channel/group) priority with an active request is selected, but the lowest numbered (channel/group) with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

24.5.3 DMA arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

24.5.3.1 Fixed group arbitration, fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller —that is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

24.5.3.2 Round-robin group arbitration, fixed channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with an service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round-robin algorithm will select the highest pending request from the next group in the round-robin sequence. Servicing continues round-robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round-robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

24.5.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in Round-robin group arbitration, fixed channel arbitration, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round-robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round-robin manner, any channel that generates DMA requests faster than a combination of the group round-robin service rate and the channel service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round-robin/round-robin mode.

24.5.3.4 Fixed group arbitration, r ound-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, c hannels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, fixed channel arbitration but all the channels in the highest priority group will get serviced. Service latency will be short on the highest priority group, but could potentially get very much longer and longer as the group priority decreases.

24.5.4 DMA transfer

This section discusses how to perform DMA transfers with the eDMA.

24.5.4.1 Single request

To perform a simple transfer of *n* bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop, transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit-wide port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
```

```
TCDn_CSR[START] = 1 (Should be written last after all other fields have been
                initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

24.5.4.2 Multiple requests

Besides transferring 32 bytes via two hardware requests, the next example is the same as previous. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.

8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware (eDMA peripheral) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
    b. Write 32 bits to location 0x2010 → first iteration of the minor loop.
    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
    d. Write 32 bits to location 0x2014 → second iteration of the minor loop.
    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
    f. Write 32 bits to location 0x2018 → third iteration of the minor loop.
    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
    h. Write 32 bits to location 0x201C → last iteration of the minor loop ® major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2(TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

24.5.4.3 Modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 16-byte size queue.

Table 24-4. Modulo feature example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

24.5.5 eDMA TCD*n* status monitoring

This section discusses how to monitor eDMA status.

24.5.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Table 24-5. TCD status sequence – software activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware (peripheral) initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Table 24-6. TCD status sequence – hardware activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit. The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

24.5.5.2 Active channel TCD*n* reads

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

24.5.5.3 Preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when round-robin arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

24.5.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done → set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a 9-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel (that is, how it can use another channel's TCD) at the end of a loop.

Table 24-7. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH] Link channel number when linking at end of minor loop (current iteration)	
Link at end of major loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH] Link channel number when linking at end of major loop	

24.5.7 Dynamic programming

The following sections describe dynamic programming.

24.5.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

- Switch to round-robin channel arbitration mode, change the channel priorities, then switch back to fixed arbitration mode.
- Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to round-robin group arbitration mode, change the group priorities, then switch back to fixed arbitration mode.
2. Disable all channels, change the group priorities, then enable the appropriate channels.

24.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in the following table is recommended when executing a dynamic channel link request.

Table 24-8. Coherency model for a dynamic channel link request

Step	Action
1	Write 1b to the TCD.major.e_link bit.
2	Read back the TCD.major.e_link bit.
3	Test the TCD.major.e_link request status: • If TCD.major.e_link = 1b, the dynamic link attempt was successful. • If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

Note

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

24.5.7.3 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

Note

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

24.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in the following table may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Table 24-9. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to theTCD.d_req bit. **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.
3	Write theTCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.
5	Read back the 16 bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: • If e_sg = 1b, the dynamic link attempt was successful. • If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). • If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

24.5.7.3.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in the following table may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 24-10. Coherency model for method 2

Step	Action
1	Write 1b to theTCD.d_req bit. **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.
2	Write theTCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: • If e_sg = 1b, the dynamic link attempt was successful. • If e_sg = 0b, read the 32 bit TCD dlast_sga field. • If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). • If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

Chapter 25
Direct Memory Access Multiplexer (DMAMUX)

25.1 Introduction

25.1.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

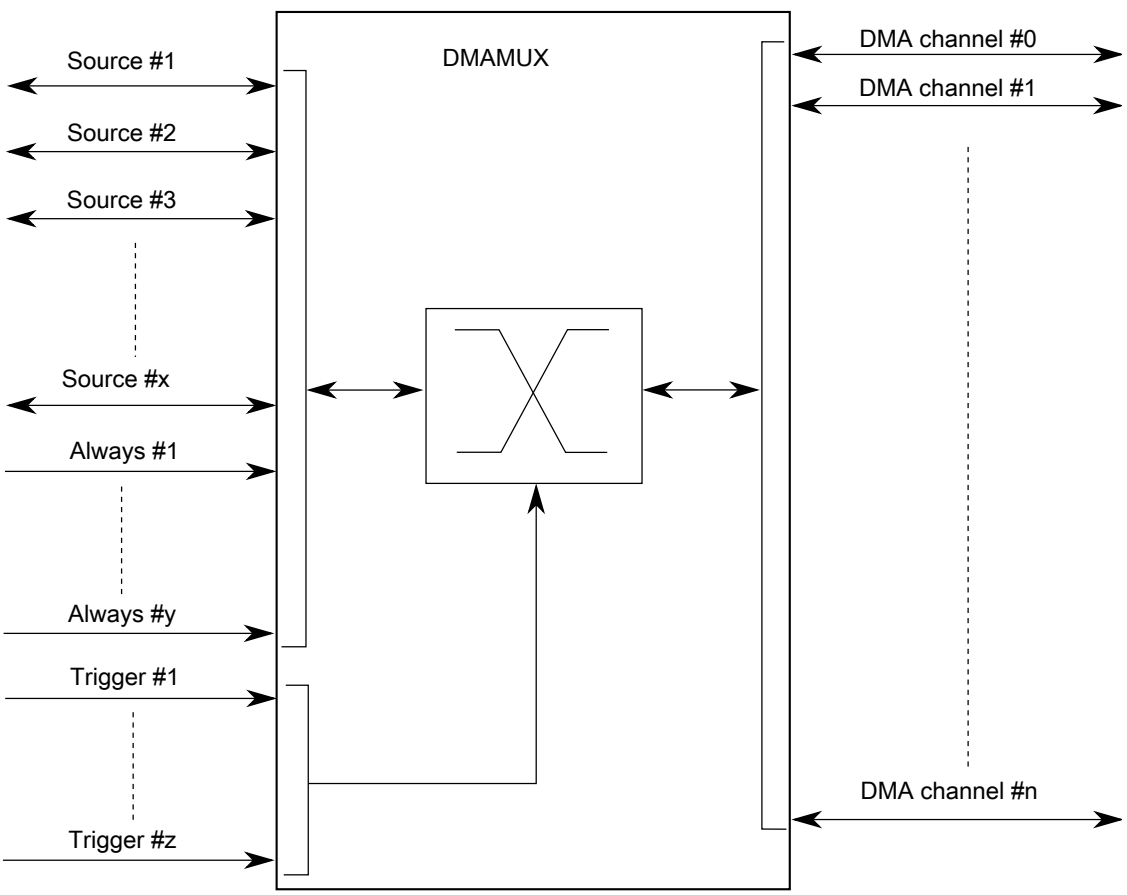


Figure 25-1. DMAMUX block diagram

25.1.2 Features

The DMAMUX module provides these features:

- Up to 64 peripheral slots and up to six always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - The first eight channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

25.1.3 Modes of operation

The following operating modes are available:

- Disabled mode

  In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically. Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–.

25.2 External signal description

The DMAMUX has no external pins.

25.3 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

DMAMUX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	25.3.1/1073
1	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	25.3.1/1073
2	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	25.3.1/1073
3	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	25.3.1/1073
4	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	25.3.1/1073
5	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	25.3.1/1073
6	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	25.3.1/1073
7	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	25.3.1/1073
8	Channel Configuration register (DMAMUX_CHCFG8)	8	R/W	00h	25.3.1/1073
9	Channel Configuration register (DMAMUX_CHCFG9)	8	R/W	00h	25.3.1/1073
A	Channel Configuration register (DMAMUX_CHCFG10)	8	R/W	00h	25.3.1/1073
B	Channel Configuration register (DMAMUX_CHCFG11)	8	R/W	00h	25.3.1/1073
C	Channel Configuration register (DMAMUX_CHCFG12)	8	R/W	00h	25.3.1/1073
D	Channel Configuration register (DMAMUX_CHCFG13)	8	R/W	00h	25.3.1/1073
E	Channel Configuration register (DMAMUX_CHCFG14)	8	R/W	00h	25.3.1/1073
F	Channel Configuration register (DMAMUX_CHCFG15)	8	R/W	00h	25.3.1/1073

25.3.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable Enables the DMA channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel. 0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
2–7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

25.4 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

25.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention. The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

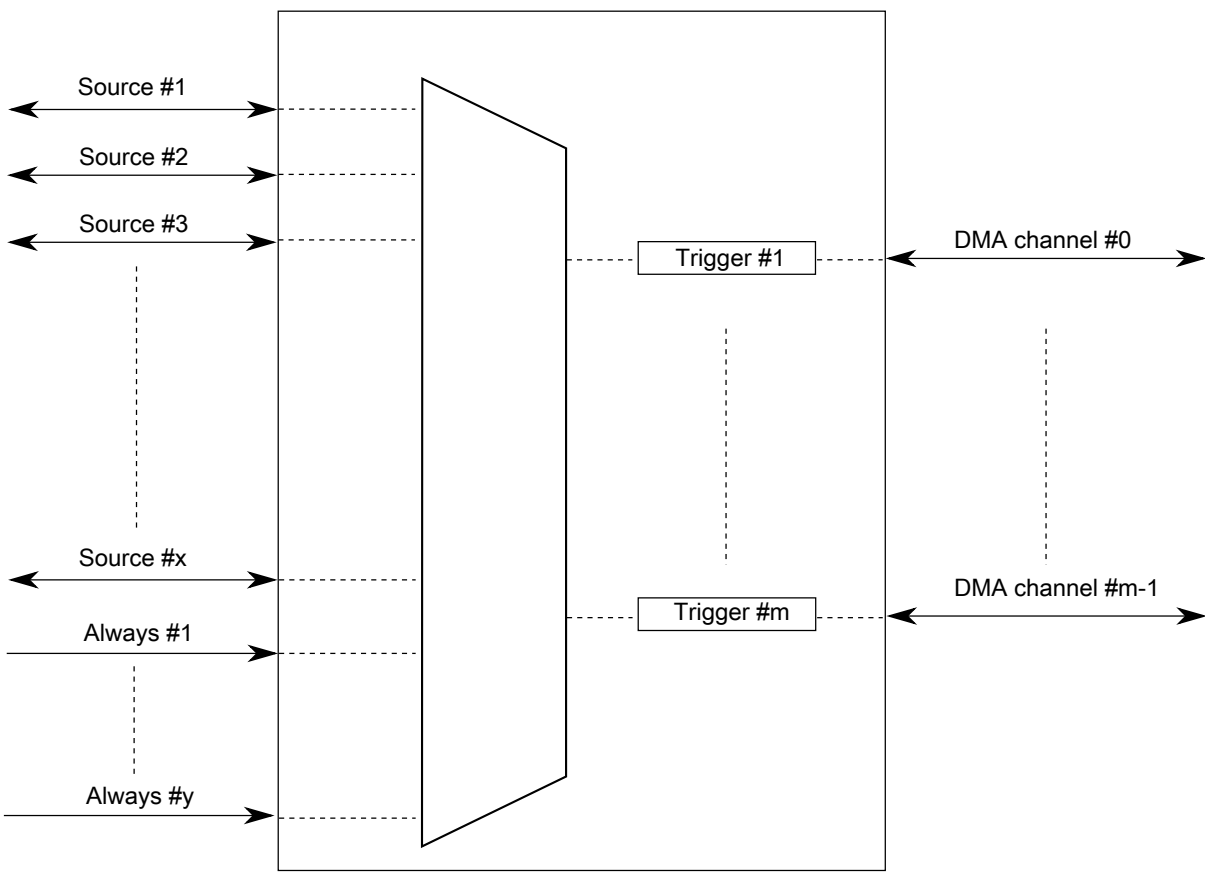


Figure 25-2. DMAMUX triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

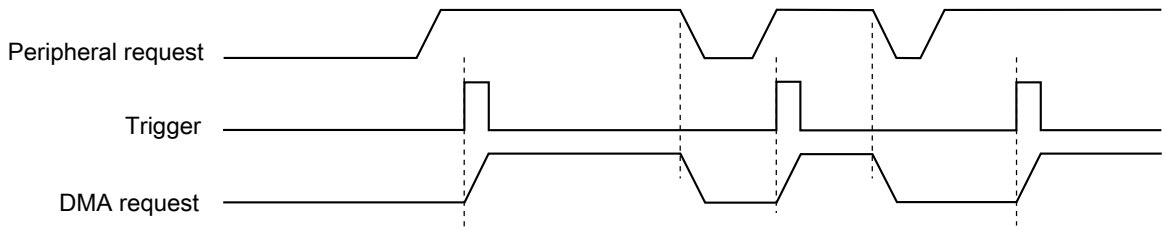


Figure 25-3. DMAMUX channel triggering: normal operation

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

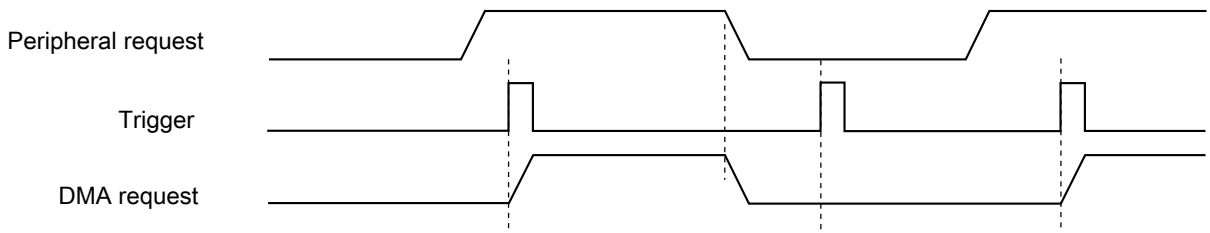


Figure 25-4. DMAMUX channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 μs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

25.4.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

25.4.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are six additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

25.5 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

25.5.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

25.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).
2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1 (base address + 0x01).

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source, without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1 (base address + 0x01).

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8 (base address + 0x08).
3. Write 0x87 to CHCFG8 (base address + 0x08). (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 26
Clocking

26.1 Introduction

This chapter describes the architecture of the system level clocks and includes the following information:

- System clock specifics.
- Clock architecture.
- Clock sources.
- Clock monitoring.
- Programmable clock dividers.
- Clock control registers.
- Progressive Clock Switching (PCS).

Note

This chapter only covers clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The system clock (CLKOUT) services the computational shell (includes the main CPUs Main Core_0 and Checker Core_0s), modules that run in the high-speed domain, memories and debug logic. Its independence from the peripheral clocks allows CLKOUT to be a frequency modulated (FM) clock with reduced electromagnetic emissions while having a precise clock for the peripheral timer and communication functions. It can be reduced during low computational activity periods to lower power consumption while providing a fixed frequency to the communication interfaces and timers.

The internal 16 MHz RC oscillator (IRCOSC) services device boot up and may be enabled via the Mode Entry Module as a backup clock in the event of PLL or external oscillator failure. See the “Mode Entry Module (MC_ME)” chapter for details.

The alternative clock sources are:

- External oscillator/crystal (XOSC).
- Internal 16 MHz RC oscillator (IRCOSC).
- PLL0 (PLL0_PHI).
- PLL1 (PLL1_PHI).

Auxiliary Clock Selectors in the Clock Generation Module (MC_CGM) allow developers to select an independent clock source for each system peripheral. There is an additional System Clock Selector used exclusively for the system clock.

The outputs of the module clock selectors have individual dividers that can divide the clock selector outputs by up to 512 for a given peripheral.

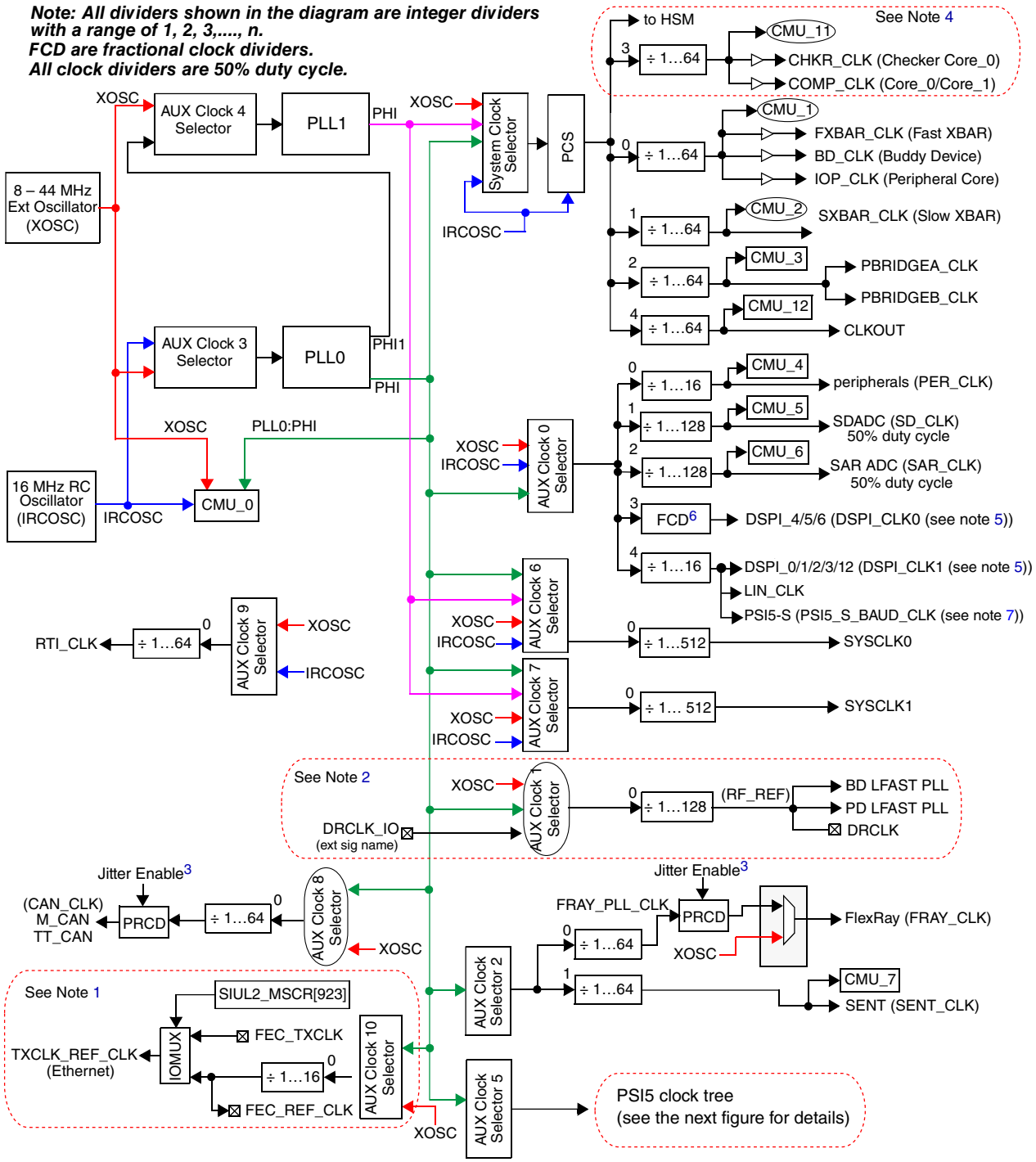
Clock Monitor Units (CMUs) are connected to outputs of clock dividers to verify that clock frequencies stay within operating limits and can be configured to signal an interrupt or initiate a system reset when an issue with a clock is found.

26.2 Clock generation

The top-level clock generation architecture can be seen in Figure 26-1. See the Clock Generation Module (MC_CGM) chapter for details on the Fractional Clock Dividers. All clock selectors and dividers are located and programmed in the MC_CGM, with the exception of the Hardware Security Module (HSM) clock divider. All dividers are integer dividers (1, 2, 3, …, *n*) with the ranges shown in the divider blocks in Figure 26-1. All dividers connected to the System Clock Selector must have 50% duty cycle outputs.

Note

Clock Monitoring Units CMU1 to CMU12 in Figure 26-1 use IRCOSC as their reference clock. The user can select either IRCOSC or XOSC as the reference clock for CMU_0.



1. See the "Device Ethernet clocking" figure for Ethernet clocking details
2. See the "Device LFAST clocking" figure for LFAST clocking details
3. See the “Password and Device Security Module (PASS)” chapter for Jitter Enable details.
4. The COMP_CLK and CHKR_CLK clock trees are isolated downstream of System Clock Divider 3. Isolation is maintained in the phyiscal design of the clock trees.
5. DSPI_CLK1 and DSPI_CLK0 are the protocol clocks as described in the “Deserial Serial Peripheral Interface (DSPI)” chapter.
6. See the MC_CGM chapter for details of the Fractional Clock Divider (FCD).
7. Referred to as ipg_baud_clk in the PSI5-S chapter.

Figure 26-1. Clock generation (part 1 of 2)

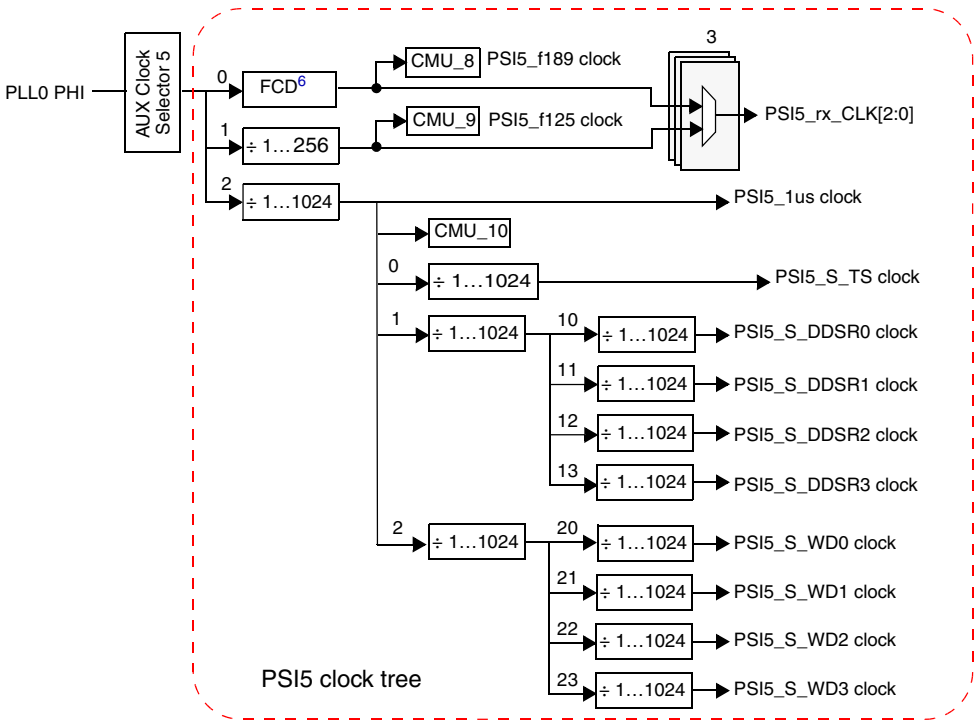


Figure 26-2. Clock generation (part 2 of 2)

26.2.1 MC_CGM registers

Table 26-1 shows the relationship between the clocks in Figure 26-1 and the MC_CGM registers.

See the Clock Generation Module (MC_CGM) for specifics on register implementation.

Table 26-1. MC_CGM relationship to clocks

MC_CGM register(s)	Register description	Output clock	CMU[1]
CGM_SC_DC0	System Clock Divider Configuration 0	Peripheral Core_2 (IOP_CLK) Fast XBAR (FXBAR_CLK) Buddy Device (BD_CLK)	CMU_1
CGM_SC_DC1	System Clock Divider Configuration 1	Slow XBAR (SXBAR_CLK)	CMU_2
CGM_SC_DC2	System Clock Divider Configuration 2	PBRIDGEA_CLK PBRIDGEB_CLK	CMU_3
CGM_SC_DC3	System Clock Divider Configuration 3	Checker Core_0s (CHKR_CLK) Main Core_0 (COMP_CLK) Main Core_1 (COMP_CLK)	CMU_11
CGM_SC_DC4	System Clock Divider Configuration 4	CLKOUT	CMU_12
CGM_AC0_DC0 CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	Peripherals (PER_CLK)	CMU_4

Table continues on the next page...

Table 26-1. MC_CGM relationship to clocks (continued)

MC_CGM register(s)	Register description	Output clock	CMU[1]
CGM_AC0_DC1 CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	SDADC (SD_CLK)	CMU_5
CGM_AC0_DC2 CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	SARADC (SAR_CLK)	CMU_6
CGM_AC0_DC3 CGM_AC0_SC	Aux Clock 0 Divider Configuration 3 Aux Clock 0 Select Control	DSPI_4/5/6 (DSPI_CLK0)	-
CGM_AC0_DC4 CGM_AC0_SC	Aux Clock 0 Divider Configuration 4 Aux Clock 0 Select Control	DSPI_0/1/2 (DSPI_CLK1) LINFLEX (LIN_CLK) PSI5-S (PSI5_S_BAUD_CLK)	-
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	BD LFAST PLL PD LFAST PLL DRCLK	-
CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	FlexRay (FRAY_CLK)	-
CGM_AC2_DC1	Aux Clock 2 Divider Configuration 1	SENT (SENT_CLK)	CMU_7
CGM_AC3_SC	Aux Clock 3 Select Control	PLL0	-
CGM_AC4_SC	Aux Clock 4 Select Control	PLL1	-
CGM_AC5_DC0	Aux Clock 5 Divider Configuration 0	PSI5_f189_CLK	CMU_8
CGM_AC5_DC1	Aux Clock 5 Divider Configuration 1	PSI5_f125_CLK	CMU_9
CGM_AC5_DC2	Aux Clock 5 Divider Configuration 2	PSI5_1µs_CLK	CMU_10
CGM_AC5_CDC0	Aux Clock 5 Cascaded Divider Configuration 0	PSI5_S_TS_CLK	-
CGM_AC5_CDC1	Aux Clock 5 Cascaded Divider Configuration 1	PSI5_S_DDSR_CLK	-
CGM_AC5_CDC2	Aux Clock 5 Cascaded Divider Configuration 2	PSI5_S_WD_CLK	-
CGM_AC5_CDC10	Aux Clock 5 Cascaded Divider Configuration 10	PSI5_S_DDSR0_CLK	-
CGM_AC5_CDC11	Aux Clock 5 Cascaded Divider Configuration 11	PSI5_S_DDSR1_CLK	-
CGM_AC5_CDC12	Aux Clock 5 Cascaded Divider Configuration 12	PSI5_S_DDSR2_CLK	-
CGM_AC5_CDC13	Aux Clock 5 Cascaded Divider Configuration 13	PSI5_S_DDSR3_CLK	-
CGM_AC5_CDC20	Aux Clock 5 Cascaded Divider Configuration 20	PSI5_S_WD0_CLK	-
CGM_AC5_CDC21	Aux Clock 5 Cascaded Divider Configuration 21	PSI5_S_WD1_CLK	-
CGM_AC5_CDC22	Aux Clock 5 Cascaded Divider Configuration 22	PSI5_S_WD2_CLK	-
CGM_AC5_CDC23	Aux Clock 5 Cascaded Divider Configuration 23	PSI5_S_WD3_CLK	-
CGM_AC6_DC CGM_AC6_SC	Aux Clock 6 Divider Configuration 1 Aux Clock 6 Select Control	SYSCLK0	-

Table continues on the next page...

Table 26-1. MC_CGM relationship to clocks (continued)

MC_CGM register(s)	Register description	Output clock	CMU[1]
CGM_AC7_DC0 CGM_AC7_SC	Aux Clock 7 Divider Configuration 0 Aux Clock 7 Select Control	SYSCLK1	-
CGM_AC8_DC0 CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	M_CAN, TT_CAN (CAN_CLK)	-
CGM_AC9_DC0 CGM_AC9_SC	Aux Clock 9 Divider Configuration 0 Aux Clock 9 Select Control	RTI_CLK	-
CGM_AC10_DC0 CGM_AC10_SC	Aux Clock 10 Divider Configuration 0 Aux Clock 10 Select Control	Ethernet (FEC_REF_CLK)	-

1. CMU_0 is not shown in table since it does not monitor an output clock from the MC_CGM.

26.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks are shown in Table 26-2.

The user is responsible to verify that these values are not exceeded. See Figure 26-1 for clocking architecture overview.

Table 26-2. Maximum system level clock frequencies

Clock Selector	Nominal programmed Max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
System clock selector[1]	600	0	CMU_1 IOP core FXBAR Core_0 Core_0s Core_1	200
		1	CMU_2 SXBAR PBRIDGE[2] - GTM/DMA/CAN	100
		2	CMU_3 PBRIDGEA_CLK PBRIDGEB_CLK	50
		3	Computational Cores (Core_0, Core_0s, Core_1)	300
		4	CLKOUT	66
			HSM_CLK	100
Aux Clock 0 Selector	400	0	PER_CLK	80
		1	SD_CLK	16
		2	SAR_CLK	16
		3	DSPI_CLK0	100
		4	DSPI_CLK1 LIN_CLK PSI5_S_BAUD_CLK (PSI5-S)	100

Table continues on the next page...

Table 26-2. Maximum system level clock frequencies (continued)

Clock Selector	Nominal programmed Max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 1 Selector	400	0	LFAST PLL	26
Aux Clock 2 Selector	400	0	FRAY_CLK	40
		1	SENT_CLK	100
Aux Clock 3 Selector	See Data Sheet		PLL0	-
Aux Clock 4 Selector	See Data Sheet		PLL1	-
Aux Clock 5 Selector	400	0	PSI5_f189_CLK (FCD output)	6.048
		1	PSI5_f125_CLK	4
		2	PSI5_1us_CLK	1
		CDC0	PSI5_S_TS_CLK	25
		CDC1	PSI5_S_DDSR_CLK	25
		CDC10	PSI5_S_DDSR0_CLK	25
		CDC11	PSI5_S_DDSR1_CLK	25
		CDC12	PSI5_S_DDSR2_CLK	25
		CDC13	PSI5_S_DDSR3_CLK	25
		CDC2	PSI5_S_WD_CLK	25
		CDC20	PSI5_S_WD_CLK0	25
		CDC21	PSI5_S_WD_CLK1	25
		CDC22	PSI5_S_WD_CLK2	25
		CDC23	PSI5_S_WD_CLK3	25
Aux Clock 6 Selector	400	0	SYSCLK0	100
Aux Clock 7 Selector	400	0	SYSCLK1	100
Aux Clock 8 Selector	400	0	CAN_CLK	50
Aux Clock 9 Selector	40	0	RTI_CLK	40
Aux Clock 10 Selector	400	0	FEC_REF_CLK	50
-	-	-	FEC_TXCLK	25
-	-	-	FEC_RXCLK	25

1. System clock selection is a component of the device run mode switching process. See the Mode Entry Module (ME_ME) chapter for details.
2. Each peripheral bridge (PBRIDGE_A, PBRIDGE_B) is capable of clocking the bus interface of each peripheral slot individually at the slow XBAR frequency, or at the slow XBAR frequency divided by two. The GTM is clocked at the slow XBAR frequency, and all other peripherals on both AIPS bridges are clocked at the slow XBAR frequency divided by two.

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

Note

HSM_CLK and PBRIDGEx_CLK are not required to be at the same frequency, but the frequency of COMP_CLK must equal CHKR_CLK, and FXBAR_CLK must equal IOP_CLK.

26.3.1 JTAG frequencies

Table 26-3 shows the maximum frequencies of the JTAG interface.

Table 26-3. JTAG frequencies

Device	Configuration	e200z7 max. internal JTAG frequency	Buddy device max. RWA internal frequency	Max. external JTAG frequency	Notes
PD[1]	Unconfigured clock	—	N/A	See electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock on PD (IRC) with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
	Clock configured to 300 MHz	150 MHz			JTAG usable at 1/2 CPU frequency
ED[2]	BD only powered	N/A	22.4 MHz		BD clocked by internal 64 MHz RC clock (RCOSC64) with +/- 30% tolerance and BD RWA usable at /2 frequency
	PD and BD powered, unconfigured clock	—	—		ED clocked by internal 8 MHz RC clock (IRC) with +/- 1.5% tolerance, and BD RWA usable at /2 frequency
	PD and BD powered, PD clock configured clock to 200 MHz	100 MHz	100 MHz		100 MHz JTAG achievable using LFAST mode debug, or s/w mode JTAGM. JTAG messages are generated internally by JTAGM and stay within BD and PD domain. The external tool interface is LFAST only

1. PD = Production Die
2. ED = Emulation and Debug Device = Production Die + Buddy Die

26.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to '1' at power on reset.

26.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (FMPLL)
- External oscillator (XOSC)/External clock (EXTAL bypass)
- 16 MHz internal RC oscillator (IRCOSC)

26.5.1 PLL

The Dual PLL provides separate system and peripheral clocks as shown in Figure 26-1. The PLLs are disabled after power is cycled and must be enabled by software. The block diagram for the Dual PLL is shown in Figure 26-3.

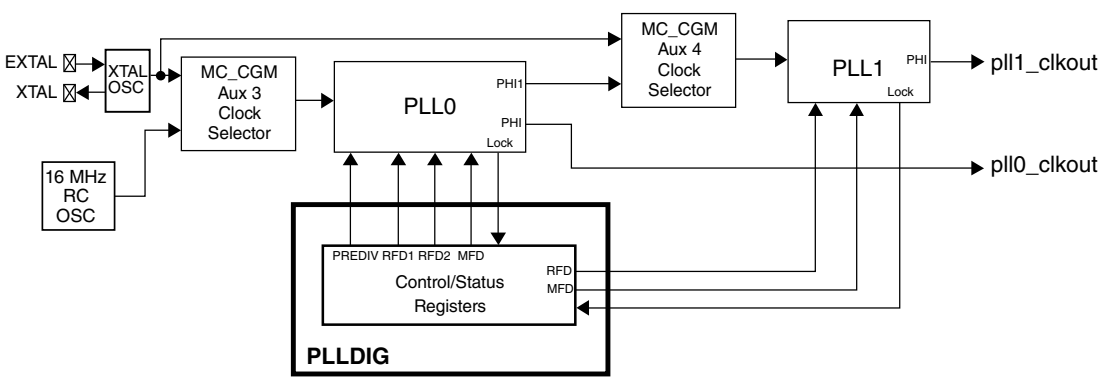


Figure 26-3. MPC5777M Dual PLL digital interface block diagram

26.5.1.1 PLL0 — base PLL (non-FM)

This primary PLL provides a clock that is not Frequency Modulated to the peripheral IPs and the reference clock to PLL1.

The input clock sources for PLL0 are:

- XOSC
- IRCOSC
- The external reference clock (input on the EXTAL pin)

The output clocks from PLL0 are:

- PHI—drives the peripheral clock and the system clock when PLL1 is not locked or active.
- PHI1—provides the input reference clock for PLL1.

26.5.1.2 PLL1 — FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

The input clocks to PLL1 are:

- PHI1 output clock from PLL0
- The external reference clock.

The output clock from PLL1 is:

- PHI—drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

26.5.1.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the "Dual PLL Digital Interface" chapter.

26.5.1.4 PLL register write protection

The PLL registers defined in Table 26-4 have write protection with:

- Soft locking—can be unlocked by software after being previously locked.
- Hard locking—can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the PLL register write protection.

Table 26-4. PLL register write protection

Offset	Register[1]	Protection
00h	PLL0 Control Register (PLL0CR)	Yes

Table continues on the next page...

Table 26-4. PLL register write protection (continued)

Offset	Register[1]	Protection
04h	PLL0 Status Register (PLL0SR)	No
08h	PLL0 Divider Register (PLL0DIVR)	Yes
20h	PLL1 Control Register (PLL1CR)	Yes
24h	PLL1 Status Register (PLL1SR)	No
28h	PLL1 Divider Register (PLL1DIVR)	Yes
2Ch	PLL1 Frequency Modulation Register (PLL1FMR)	Yes
30h	PLL1 Fractional Divider Register (PLL1FDR)	Yes

1. See Register Protection Configuration chapter for bit field details.

26.5.1.5 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in the following table. All other register reset values are shown in the Dual PLL Digital Interface chapter.

Table 26-5. Dual PLL Digital Interface register reset values

Offset (hex)	Register	Reset value (hex)
0008	PLL0DV — PLL0 Divider Register	0000_0000
0028	PLL1DV — PLL1 Divider Register	0000_0000
0034	CLKMUX — Clock Multiplexer Control Register	0000_0000
0038	Reserved	—

26.5.1.6 Loss of clock detection

Loss of clock detection should be enabled at all times when the PLLs are enabled. There is a software option to generate a reset or interrupt in the event of a loss of clock condition. Software may enable or disable an optional backup clock in the PLL.

26.5.1.7 PLLDIG initialization information

Configure PLL0 and related modules.

1. With PLL0 disabled, program PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select XOSC as source.

2. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

3. If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).

4. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

Turn on XOSC and PLL0.

5. Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<mode>_MC[XOSCON] = 1 and MC_ME_<mode>_MC[PLL0ON] = 1. If desired, also set MC_ME_<mode>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0(PLL0 PHI output) as the sysclk.

6. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

7. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

- XOSC counter expires (if the new mode configuration changes MC_ME_<mode>_MC[XOSCON] = 1), and
- PLL is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLL0ON] = 1)

8. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Configure PLL1 and related modules.

9. With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = 1 which selects XOSC.

10. Program PLL1DV[MFD] and PLL1DV[RFDPHI].

11. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

Turn on PLL1.

12. Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL1 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<mode>_MC[XOSCON] = 1, MC_ME_<mode>_MC[PLL1ON] = 1 and MC_ME_<mode>_MC[PLL1ON] = 1. If desired, also set MC_ME_<mode>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.

13. Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

14. Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

- XOSC counter expires (if the new mode configuration changes XOSCON to 1), and
- PLL1 is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLL1ON] = 1).

15. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Note: See the "Mode Entry Module (MC_ME)" chapter in this Reference Manual more details.

26.5.2 External oscillator (XOSC)

The external oscillator (XOSC):

- Is a reference for the on-chip PLLs.
- Can be used as a clock source for the ADCs, timer, serial interfaces, RTI, LFAST and as a system clock source.
- Is available for observation on either of the CLKOUT pins.
- Is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Internal load capacitors for 20 MHz/40 MHz crystal oscillators.
- Automatic Loop Control (ALC) to remove need for external series resistor.
- Gain selection for <= 20 MHz and 40 MHz crystals to allow for optimal startup margin without overdrive issues.
- Reference clock to PLL0.
- Reference clock to CMU0 (IRCOSC trimming).
- Option to drive the CAN and FlexRay protocol clocks directly from the XOSC.

The XOSC provides support for 8 – 44 MHz crystal inputs, has integrated load capacitors and ALC to remove the need for external series resistor.

26.5.2.1 XOSC Start-up

Enabling of the oscillator at startup is determined by a bit in the UTEST row of flash memory. The default value has the XOSC disabled:

- UTEST Miscellaneous[7] = 0 (XOSC EN).

If enabled, the oscillator is started in PHASE 3 of the reset sequence: after negation of the internal POR circuits but while the external PORST input is asserted. This is done to allow fast application start-up time while accounting for settling time of the power regulators.

The selection of internal or external load capacitors on the XTAL/EXTAL pins is also determined by a bit in the UTEST row of flash memory. The default value is external load capacitors:

- UTEST Miscellaneous[6] = 1 (XOSC EXT LOAD).

Load capacitor values are determined by the crystal manufacturer data sheet requirements, while accounting for stray PCB and on-chip capacitance. See the device data sheet for on-chip capacitance values.

When using the internal load capacitors for the oscillator, the startup value of the capacitors is stored in the UTEST row of the flash memory. During PHASE 3 of the device reset sequence, the oscillator UTEST values are read and driven to the oscillator. The enable/disable state of the oscillator is captured in the MC_ME module at this time. After reset, the oscillator can be enabled/disabled by software in the MC_ME module.

Internal load capacitor selection is maintained by the oscillator if disabled after reset. The default reset value for the XOSC trimming capacitors does not trim the internal capacitor values:

- UTEST Miscellaneous[5:1] = 00000b (XOSC LOAD CAP SEL).

The internal load capacitor values stored in UTEST flash memory row have triple-voting flip-flop (TVF) implementation to prevent an incorrect value causing an undetectable error in the system.

The XOSC has the ability to be started with either 8 MHz–20 MHz, or 40 MHz–44 MHz crystal. The default value of this field is for an 8 MHz–20 MHz crystal:

- UTEST Miscellaneous[10] = 0 (XOSC EN_40MHz).

See the flash memory chapter for more information on the UTEST row programming.

26.5.2.2 XOSC register write protection

The XOSC registers defined in Table 26-6 have write protection with:

- Soft locking—can be unlocked by software after being previously locked.
- Hard locking—can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the XOSC register write protection.

Table 26-6. XOSC register write protection

Offset	Register[1]	Protections?
00h	XOSC_CTL—XOSC Control Register	Yes

1. See Register Protection Configuration chapter for bit field details

26.5.2.3 XOSC reset value

The following table shows the default reset value for the XOSC registers.

Table 26-7. XOSC register reset values

Offset	Register	Reset value
00h	XOSC_CTL—XOSC Control Register	0030_8000h

26.5.3 16 MHz internal RC oscillator (IRCOSC)

The 16 MHz internal RC oscillator default clock is always enabled on reset and can be the clock source for the PLLs. The register interface is for user trimming of the oscillator and dividing output from 1 to 32.

26.5.3.1 IRCOSC register interface

A dedicated digital interface for the IRCOSC has a user register for fine tuning the IRCOSC frequency. Other read-only registers contain settings and values for the IRCOSC temperature sensor, voltage regulator and capacitor trimming.

26.5.3.2 IRCOSC reset value

Table 26-8 shows the default reset value for the IRCOSC registers.

Table 26-8. IRCOSC register reset values

Offset	Register	Reset value
00h	CTL—IRCOSC Control Register	0000_0000h
04h	NT—IRCOSC Native Trimming Register	0000_0000h
08h	TT—IRCOSC Temperature Trimming Register	0000_0000h

26.5.3.3 IRCOSC register write protection

The IRCOSC registers do not utilize write protection.

26.6 Peripheral clocks

The following figure shows the clock distribution to the cores and peripheral modules.

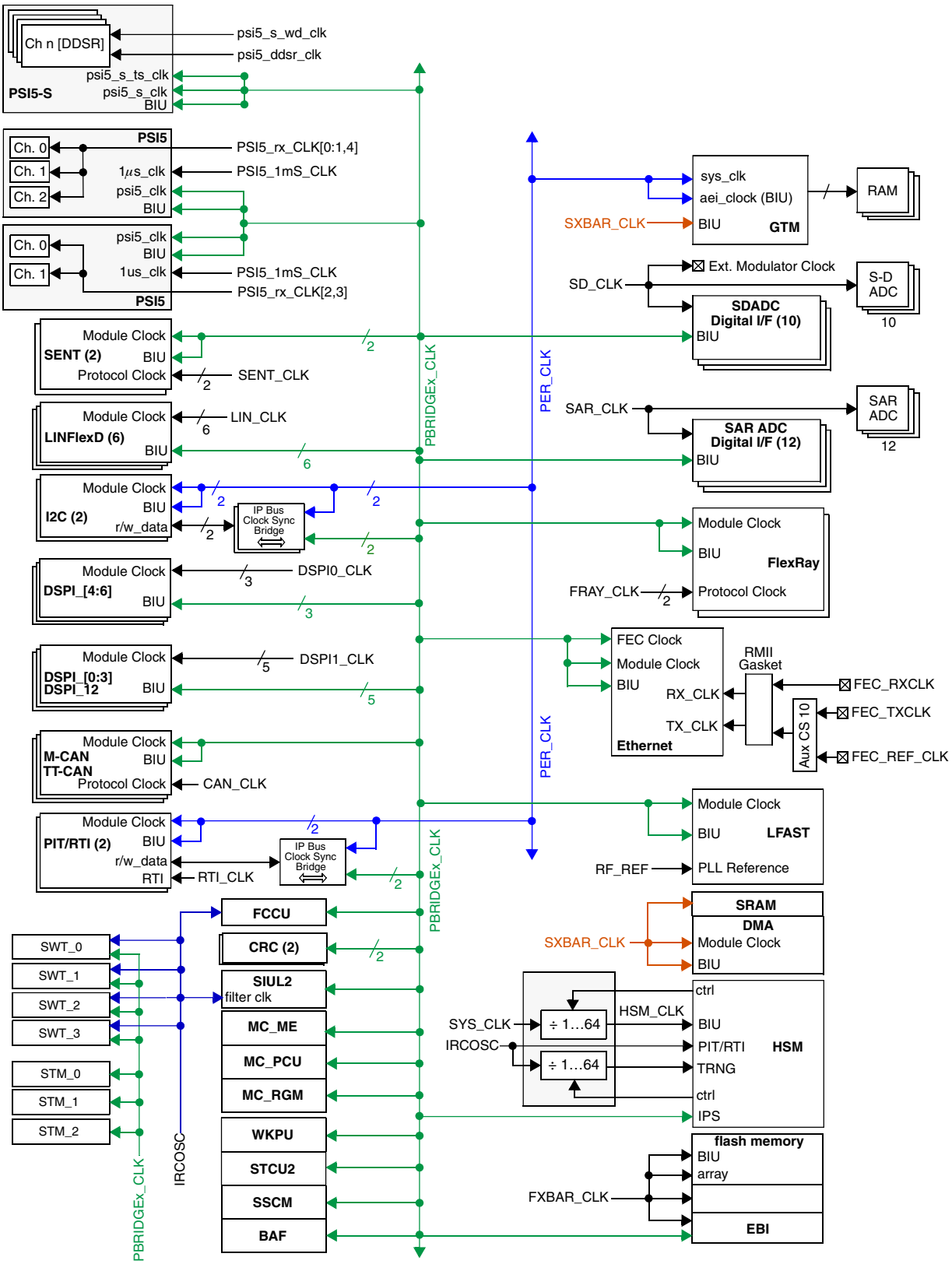


Figure 26-4. Clock distribution

26.6.1 HSM clock divider

The clock dividers for the HSM core and random number generator (TRNG) are located outside the HSM, but not in the MC_CGM. These dividers are controlled by the HSM.

The HSM contains an internal PIT/RTI block, which is clocked by the IRCOSC.

The HSM has an internal clock monitor on its input clock, so there is no need for a Clock Monitoring Unit (CMU) for the HSM_CLK in Figure 26-1.

26.6.2 PSI5 clock dividers

The MPC5777M microcontroller includes three PSI5 modules, all of which use the three common clocks shown in Figure 26-1. The required frequencies for these clocks are generated from PLL0 as follows:

- $F_{PSI5\_f189\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC0[DIV] + 1) / 10CGM_AC5_DC0[FMT]) = 6.048 MHz ±0.1%
- $F_{PSI5\_f125\_CLK} = F_{PLL0\_PHI}$ / (CGM_AC5_DC1[DIV] + 1) = 4 MHz
- $F_{PSI5\_1us\_CLK} = F_{PLL0\_PHI}$ / (CGM_AC5_DC2[DIV] + 1) = 1 MHz

The PSI5-S transceiver clock (ipp_psi5_uart_tclk in the PSI5-S chapter) is sourced from PSI5_f189_CLK. The transceiver clock operates in the range of 2 to 8 times the UART baud rate (up to 6.25 Mbit/sec).

- $F_{PSI5\_S\_UART\_TCLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC0[DIV] + 1) / 10CGM_AC5_DC0[FMT])
- 12.4 MHz < $F_{PSI5\_S\_UART\_TCLK}$ < 50 MHz

CAUTION

Since the PSI5 modules and the PSI5-S module use the PSI5_f189_CLK clock at significantly different frequencies, the PSI5 modules and PSI5-S module should not be used simultaneously.

The remaining PSI5-S clocks are generated by dividing the PSI5_1µs_CLK further via a second and a third level of 'cascaded' clock dividers. The individual frequencies are, therefore, derived from the PLL0 PHI frequency as follows:

- $F_{PSI5\_S\_TS\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC0[DIV]+1))

- $F_{PSI5\_S\_DDSR0\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC10[DIV]+1))
- $F_{PSI5\_S\_DDSR1\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC11[DIV]+1))
- $F_{PSI5\_S\_DDSR2\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC12[DIV]+1))
- $F_{PSI5\_S\_DDSR3\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC13[DIV]+1))
- $F_{PSI5\_S\_WD0\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC20[DIV]+1))
- $F_{PSI5\_S\_WD1\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC21[DIV]+1))
- $F_{PSI5\_S\_WD2\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC22[DIV]+1))
- $F_{PSI5\_S\_WD3\_CLK} = F_{PLL0\_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC23[DIV]+1))

26.6.3 LFAST clocking

The MPC5777M includes two LFAST modules:

- In conjunction with the SIPI module for interprocessor communications.
- In conjunction with the JTAGM module for high speed debug.

High-speed operation of both LFAST modules is supported by a single LFAST PLL requiring a 10–20 MHz reference. Low-speed LFAST operation uses the reference clock directly.

The LFAST PLL reference clock source may be:

- PLL0:PHI output.
- External oscillator (XOSC).
- Input from the external LFAST device via the DRCLK_IO package pin.

When the reference clock is generated internally, it can be output on the DRCLK_IO pin.

The debug LFAST module only supports the operation mode where it outputs the reference clock to a connected LFAST capable tool as the DRCLK pin only supports output of the DRCLK reference.

On Emulation and Debug Devices (ED), the Buddy Device (BD) LFAST module connects to the package debug pins in place of the Production Device (PD) debug LFAST module. This module has a separate, dedicated LFAST PLL and the DRCLK pin only supports output of the reference clock to a connected LFAST capable tool (as with the Production Device).

Figure 26-5 shows the clock routing for the various LFAST modules on the device, including connections to the DRCLK_IO and DRCLK pins.

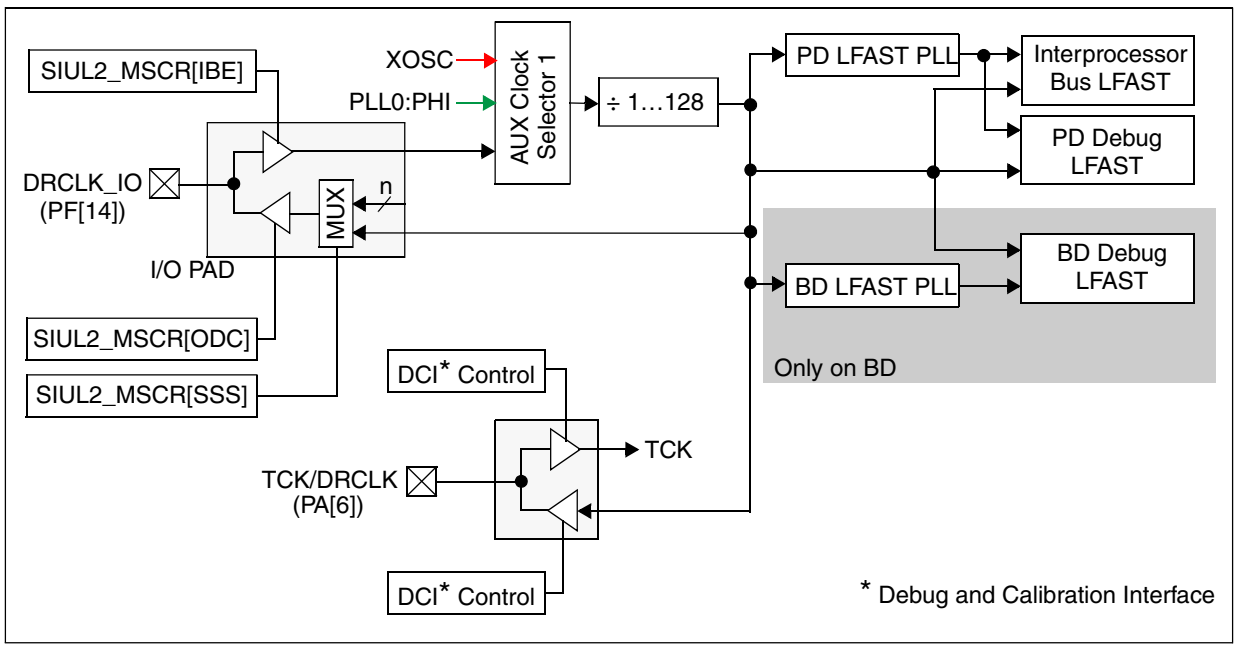


Figure 26-5. Device LFAST clocking

26.6.4 Ethernet clocking

Both internal and external clocks sources for MII and RMII Ethernet interface modes are supported. The MII clocks (FEC_TXCLK/FEC_RXCLK) are 25 MHz, and the RMII clock (FEC_REF_CLK) is 50 MHz. The following figure gives the connections for the Ethernet clocks.

The RXCLK pin is an input used in MII mode only and ignored by the RMII gasket in RMII mode.

The FEC_TXCLK pin is an input used in MII mode only but is shared with the FEC_REF_CLK input of the RMII gasket. The Aux clock selector is therefore required to select between FEC_TXCLK and FEC_REF_CLK for MII and RMII modes.

The FEC_REF_CLK can be generated internally or input to the device. For MII mode, the FEC_REF_CLK can be used for the 25 MHz reference for the external Ethernet PHY, which provides the TXCLK/RXCLK clocks back to the device. For RMII m ode, FEC_REF_CLK can be generated internally and fed to both the PHY device and to the RMII gasket by enabling the FEC_REF_CLK pad input and output buffers. FEC_REF_CLK can also be provided externally as an input to the device.

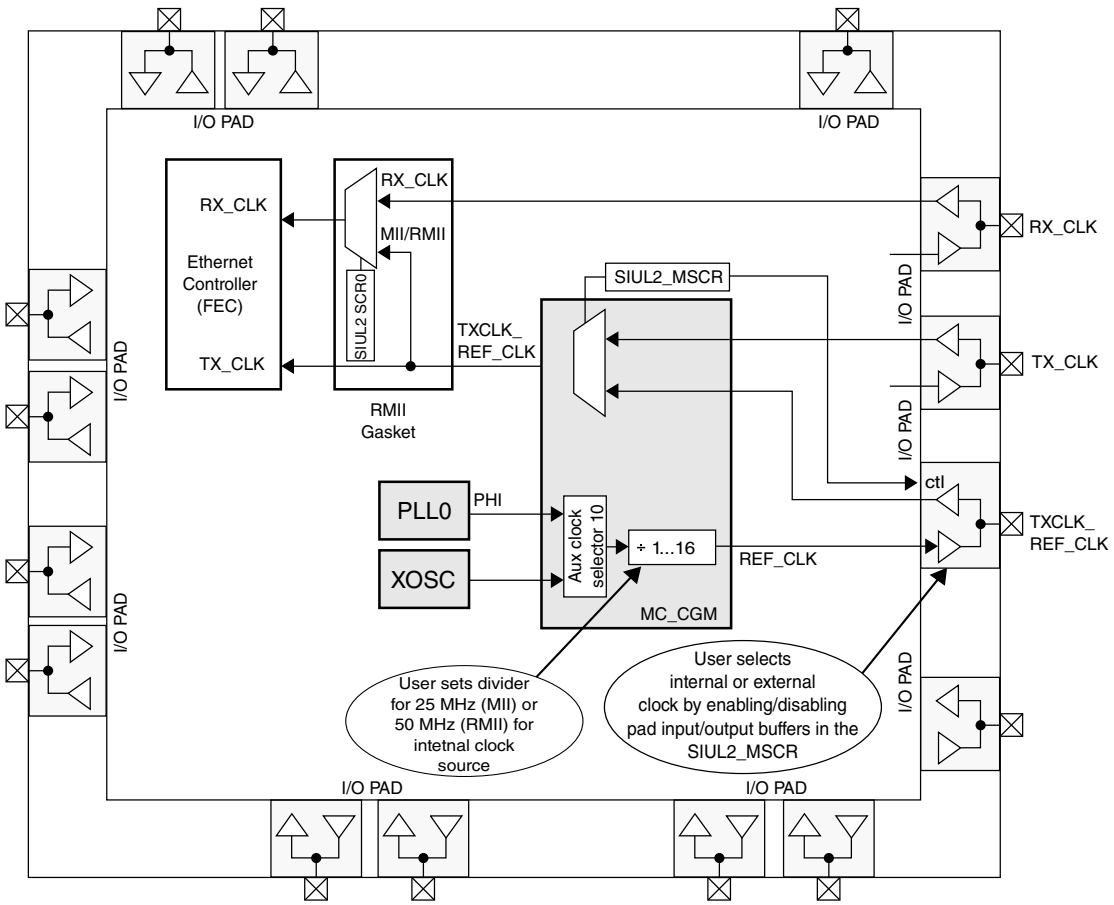


Figure 26-6. Device Ethernet clocking

26.6.5 FlexRay clocking

The FlexRay protocol clock is sourced from the PLL0_PHI clock or the external oscillator (XOSC), but its source selection is different from other modules:

1. PLL0_PHI enters the MC_CGM where it is divided by a value configured in the CGM_AC2_DC0 register and output as FRAY_PLL_CLK.
2. FRAY_PLL_CLK passes through the Pseudo Random Clock Divider (PRCD) which:

- Divides FRAY_PLL_CLK by 16.
- Adds approximately 40% of random jitter to both edges of the clock.

3. An internal multiplexer selects the PRCD output or XOSC as the resultant FRAY_CLK.

PLL0_PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

When XOSC is the input clock to the FlexRay controller, a 40 MHz crystal is required for proper operation. Both edges of the clock are used by the module to achieve the required 80 MHz frequency.

Selection of PLL or XOSC as the FlexRay clock is shown in Figure 29-7 in the Clock Generation Module (MC_CGM) chapter.

26.6.6 M_TTCAN/M_CAN clocking

The CAN_CLK is configured in the CGM via the following registers:

- CGM_AC8_SC selects the XOSC or PLL0:PHI as the clock source.
- CGM_AC8_DC0 reduces the CAN_CLK by configuring the dividing factor from 1 to 64.

CAN_CLK passes through the Pseudo Random Clock Divider (PRCD) which:

- Divides CAN_CLK by 16.
- Adds approximately 40% of random jitter to both edges of the clock.

PLL0.PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

26.6.7 Sigma-Delta ADC clocking

The analog portion of the Sigma-Delta ADC (SDADC) works with either an internal or external modulator.

When the internal modulator in the SDADC in enabled the internal clock (SD_CLK) is always used, see Figure 26-4).

When the external modulator is enabled (by disabling the internal modulator), an SDADC internal mux selects either:

- An external input clock (BS_CK).
- An internally generated output clock (SD_CLK) connected to the external modulator clock package pin (BS_CK).

In order to provide an internal clock option when the internal modulator is enabled, the SD_CLK output from the MC_CGM must be connected to the BS_CK pin as shown in Figure 26-7.

The MSCR in the SIUL2 control of the input/output direction of the BS_CK pin.

Each SDADC is connected as shown in Figure 26-7, with the SD_CLK being common to all modules, but each SDADC having a dedicated BS_CK pin.

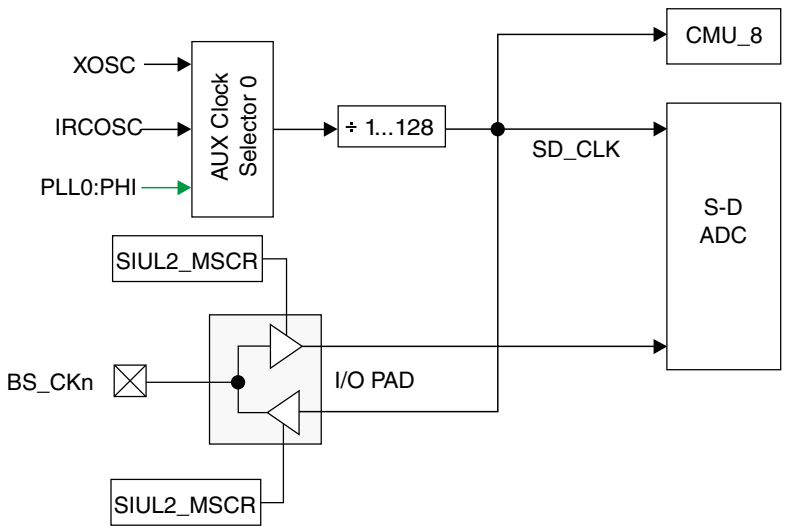


Figure 26-7. Sigma-Delta ADC clocking

26.6.8 System Clock/GTM output clock/Ethernet clock pin muxing

The two system clocks (SYSCLK[1:0]) have an independant source clock selected from:

- PLL0_PHI and PLL1_PHI.
- Internal oscillator (IRCOSC)
- External oscillator (XOSC)

The registers CGM_AC6_SC and CGM_AC7_SC configure the clocks for SYSCLK0 and SYSCLK1 respectively. See Figure 26-1.

There are three GTM output clocks available on package pins, EXTCLK[2:0]:

- EXTCLK[1:0] are muxed on the same pins as SYSCLK[1:0].
- EXTCLK[2] is muxed on the same pin as the Ethernet RMII clock (FEC_REF_CLK).

The selection of EXTCLK[n], SYSCLK[n] or FEC_REF_CLK is handled in the SIUL2_MSCR registers. See Figure 26-8.

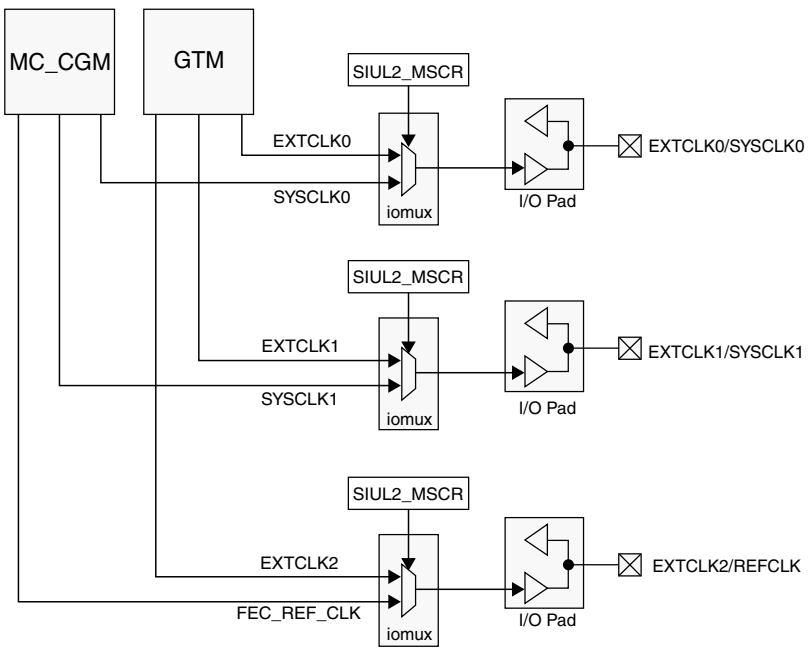


Figure 26-8. SYSCLK/EXTCLK/REF_CLK pin muxing

26.7 Clock monitoring

For all safety relevant clocks the microcontroller uses clock monitoring units (CMUs) to detect a missing clock or incorrect frequency.

Each CMU is programmed independently (see Figure 26-1 for CMU distribution) and uses the IRCOSC or XOSC as the clock monitor reference.

Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

26.7.1 CMU configuration

This section explains the CMU configuration.

Figure 26-9 shows the block diagram for CMU0 on the MPC5777M.

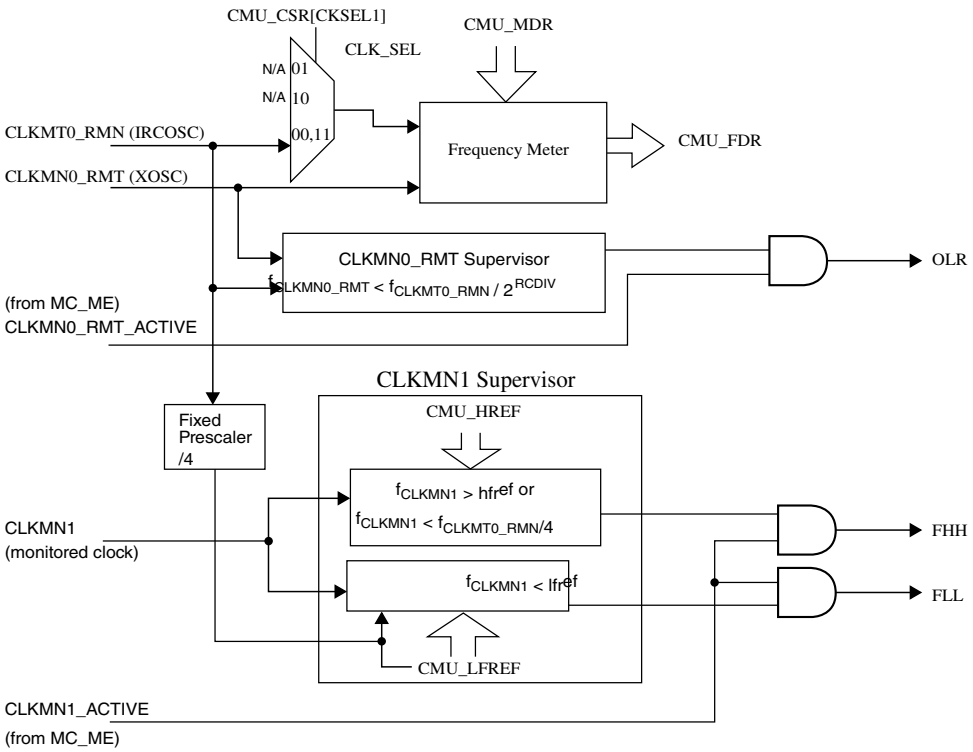


Figure 26-9. CMU0 block diagram

Figure 26-10 shows the block diagram for CMU[1:12] on the MPC5777M.

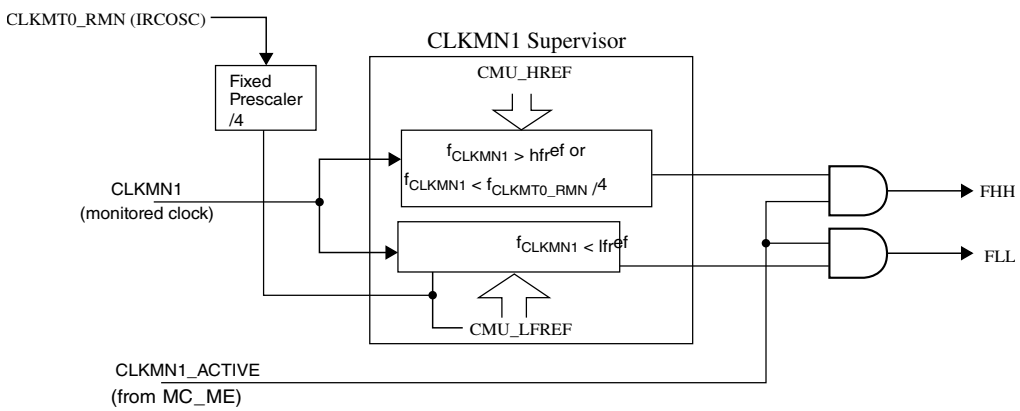


Figure 26-10. CMU[1:12] block diagram

26.7.1.1 Clock input sources

The following table shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU0 implements the XOSC monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 26-9. Clock input sources

Clock module	Monitored clock
CMU0	PLL0:PHI, XOSC, IRCOSC
CMU1	Peripheral Core_2, FXBAR_CLK, BD_CLK
CMU11	Main Core_1, Safety Core (Main Core_0, Checker Core_0s)
CMU12	CLKOUT

26.7.1.2 CMU registers and field availability

Table 26-10 specifies which registers and fields are available for a given CMU.

Table 26-10. CMU register availability

Address offset	Register	CMU	Note
0000h	CMU_CSR	All	CMU_CSR[SFM] and CMU_CSR[CKSEL1] in CMU0 only.
0004h	CMU_FDR	CMU0	—
0008h	CMU_HFREFR	All	—
000Ch	CMU_LFREFR	All	—
0010h	CMU_ISR	All	CMU_ISR[OLRI] in CMU0 only.
0014h	Reserved	—	—
0018h	CMU_MDR	CMU0	—

26.7.1.3 CMU register write protection

The CMU registers defined in Table 26-11 have write protection with:

- Soft locking—can be unlocked by software after being previously locked.
- Hard locking—can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the CMU register write protection.

Table 26-11. CMU register write protection

Offset	Register[1]	Protection
0000h	CMU Control Status Register (CMU_CSR)	Yes
0004h	CMU Frequency Display Register (CMU_FDR)	No
0008h	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	No
000Ch	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	No
0010h	CMU Interrupt Status Register (CMU_ISR)	No
0014h	Reserved	No
0018h	CMU Measurement Duration Register (CMU_MDR)	No

1. See Register Protection chapter for bit field details.

26.7.2 PLL0 monitor

Software can program an upper and lower limit on the expected PLL0 PHI output clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set and an interrupt, if enabled, is generated. The default condition of the clock monitor is disabled.

26.7.3 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set and an interrupt, if enabled, is generated.

26.7.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

26.7.5 System clock monitors

A CMU is assigned to monitor the frequency of the computational core, IOP core, fast and slow XBAR, both peripheral bridges, peripheral, PSI5, SENT and ADC clocks (refer to Figure 26-1).

26.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. Figure 26-11 gives a high-level view of the loss of clock logic.

26.8.1 Loss of PLL/XOSC clock

As shown in Figure 26-11, each loss of lock signal from each PLL and the XOSC failure signals are monitored by the FCCU.

When a clock failure occurs, the FCCU can be programmed to generate:

- An interrupt: only the backup clock in the PLL can provide clocks to the system in order to perform a system clock switch. There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).
- A short reset sequence: the clock configuration is maintained but also in this situation the PLL backup clock is the only available clock to clock the system through an MC_ME mode switch.
- A long reset sequence: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states and the system clock is switched to the IRCOSC

26.8.2 Loss of IRCOSC clock

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot and backup clock, a failure is a catastrophic failure.

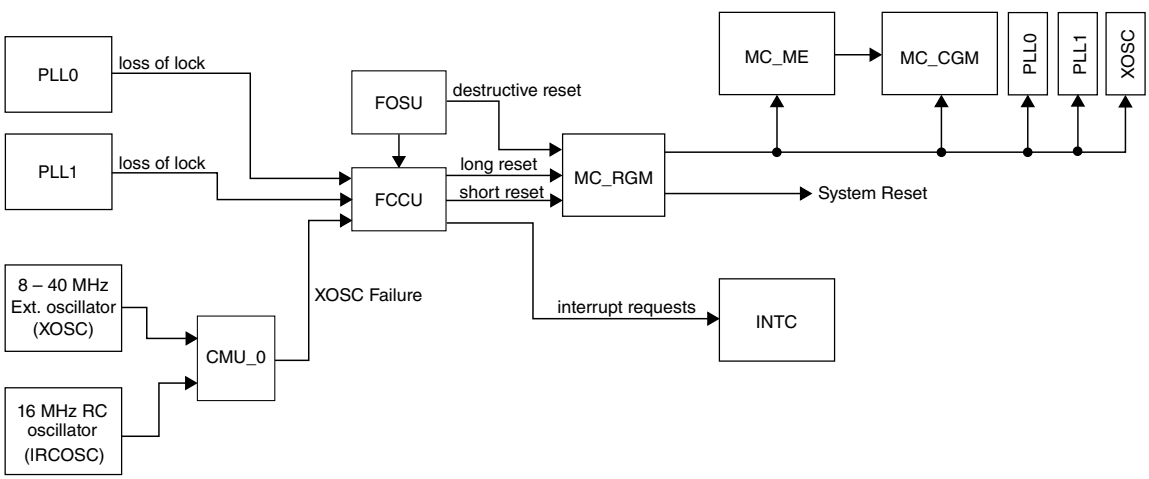


Figure 26-11. Loss of clock logic

26.9 Progressive clock switching

The Progressive Clock Switching (PCS) blocks are shown in the system-level clock diagram in Figure 26-1. They are used for ramping the system and peripheral clocks and for obtaining the precise clock frequency input for the PSI5 module. See the clock Generation Module (MC_CGM) chapter for more information on the PCS.

Chapter 27
Dual PLL Digital Interface (PLLDIG)

27.1 Introduction

The Dual PLL system composed of PLL0 and PLL1 analog blocks and the digital interface (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1.

A key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1.

27.2 Block Diagram

Please refer to the “Clocking chapter” of this *Reference Manual* to see the block diagram.

27.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for decreasing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.

- The loss of lock indication is sent to the FCCU via the system glue logic.

27.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG].

NOTE

Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<*mode*>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLLs. When using a crystal for the reference clock, reset should remain asserted until the oscillator has stabilized.

27.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In normal mode, PLL0 receives an input clock from the reference which can be prescaled by the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0 PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, and modulation enable are programmed by configuring the PLLFM register.

NOTE

While powering down the PLLs and if PLL0:PHI1 is used as the source for PLL1, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Configuring the appropriate MC_ME_<*mode*>_MC register to start PLL1 should only be done after PLL0 has achieved lock.

27.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

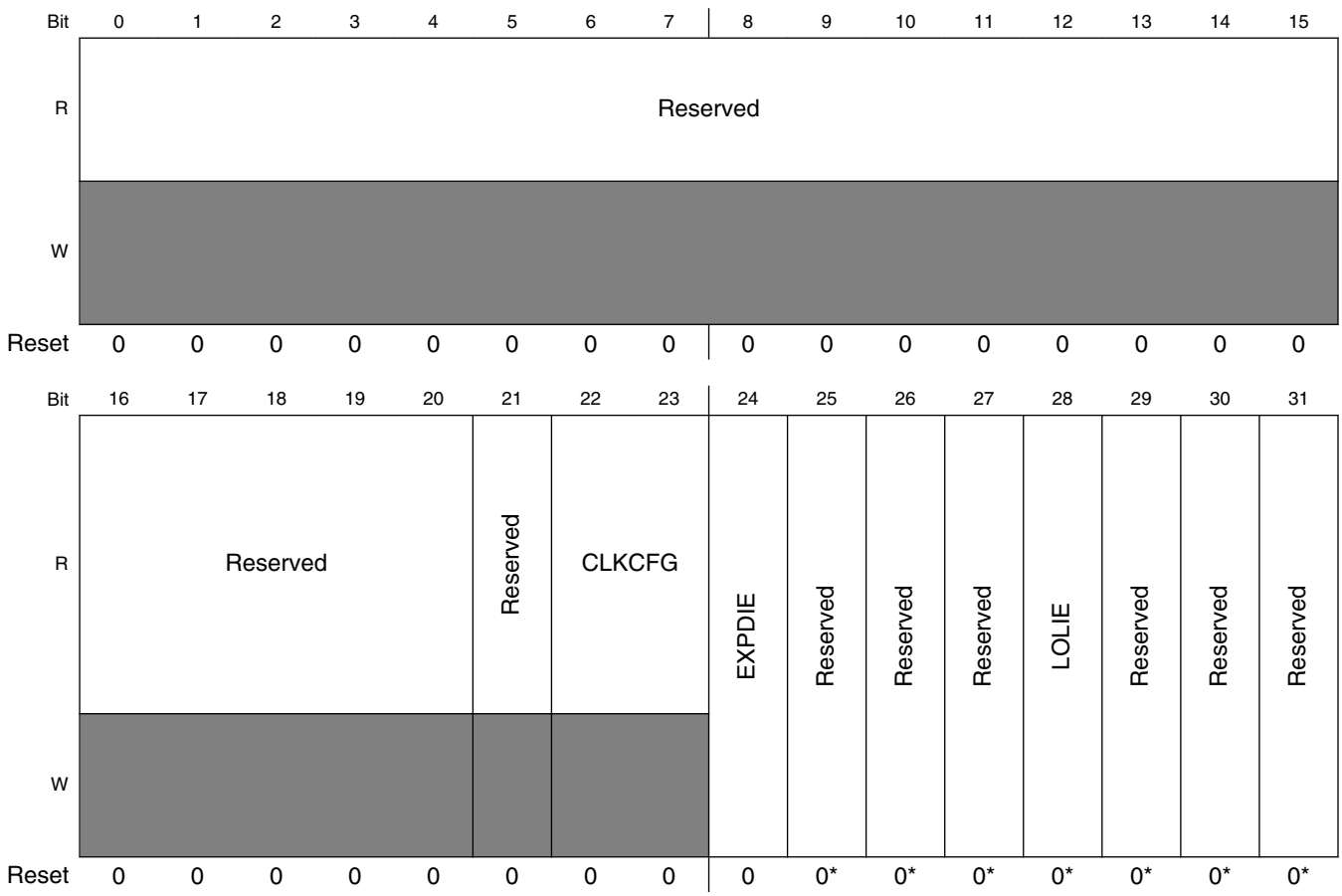
PLLDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)	32	R/W	See section	27.5.1/1114
4	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)	32	R/W	See section	27.5.2/1116
8	PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)	32	R/W	See section	27.5.3/1118
20	PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)	32	R/W	See section	27.5.4/1120
24	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)	32	R/W	See section	27.5.5/1122
28	PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)	32	R/W	See section	27.5.6/1123
2C	PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)	32	R/W	See section	27.5.7/1125
30	PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)	32	R/W	0000_0000h	27.5.8/1127

27.5.1 PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)

The PLL0CR is shown in the following table.

Address: 0h base + 0h offset = 0h



* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: See the "Clocking" chapter for details on configuring this field.

PLLDIG_PLL0CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of PLL.

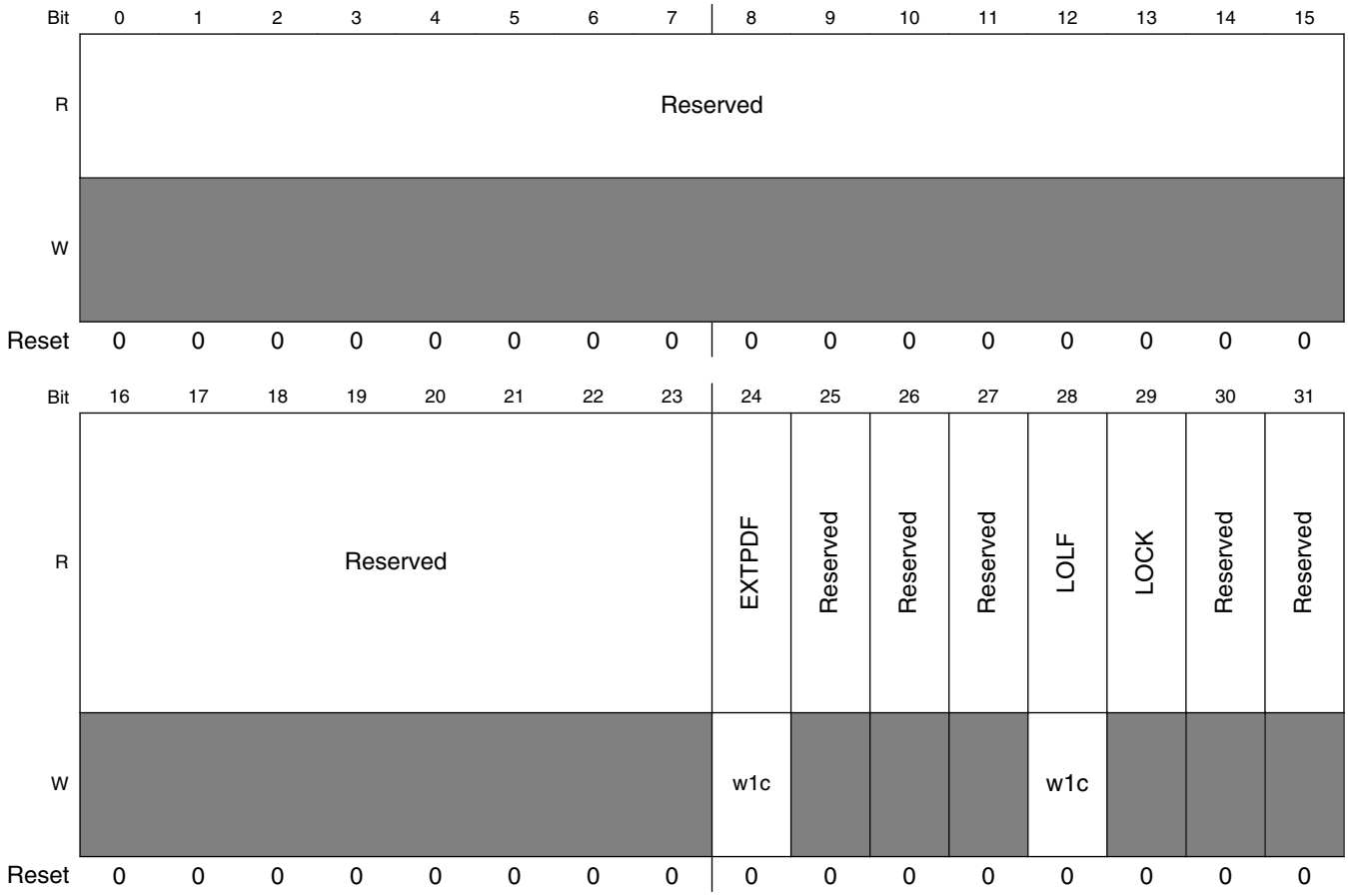
Table continues on the next page...

PLLDIG_PLL0CR field descriptions (continued)

Field	Description
	When PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1. **NOTE:** CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable. This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

27.5.2 PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)

Address: 0h base + 4h offset = 4h



PLLDIG_PLL0SR field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.

Table continues on the next page...

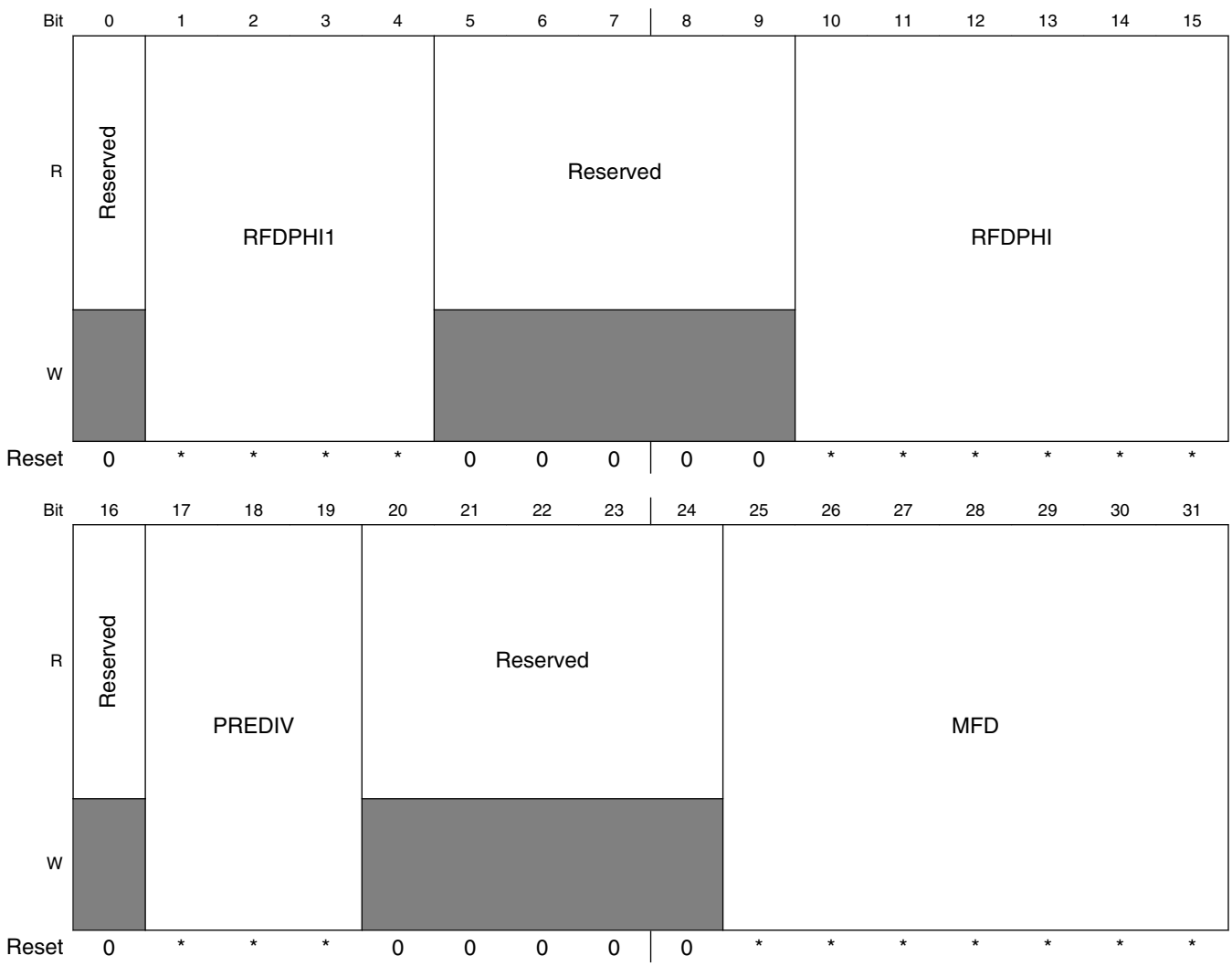
PLLDIG_PLL0SR field descriptions (continued)

Field	Description
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

27.5.3 PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider. PLL0DV can be modified at anytime, but the changes become effective only after the PLL is disabled, then reenabled. If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or interrupt based on which is enabled. The reduced frequency divider bitfields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective when PLL0 is disabled, then reenabled.

Address: 0h base + 8h offset = 8h



* Notes:
- MFD field: See Clocking chapter for reset value
- PREDIV field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

- RFDPHI1 field: See Clocking chapter for reset value

PLLDIG_PLL0DV field descriptions

Field	Description
0 Reserved	This field is reserved.
1–4 RFDPHI1	PHI1 reduced frequency divider. This 4-bit field determines the VCO clock post divider for driving the PHI1 output clock. RFDPHI1 has a range of 4..15 (4h..Fh). All other values are reserved.
5–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16 Reserved	This field is reserved.
17–19 PREDIV	Input clock predivider. This field controls the value of the divider on the input clock. The output of the predivider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 Divide by 6 111 Divide by 7
20–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 8...127 (8h...7Fh). All other values are reserved (see Clock configuration for details).

27.5.4 PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)

The PLL1CR is shown in the following table. It contains the operational state of PLL1, and is used to enable/disable PLL specific interrupts.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					Reserved	CLKCFG		EXPDIE	Reserved	Reserved	Reserved	LOLIE	Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0*	0	0	0	0*	0*	0*	0*	0*	0*	0*

* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: Please refer to device "Clocking" chapter for details on configuring this field.

PLLDIG_PLL1CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of the PLL. When the PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.

Table continues on the next page...

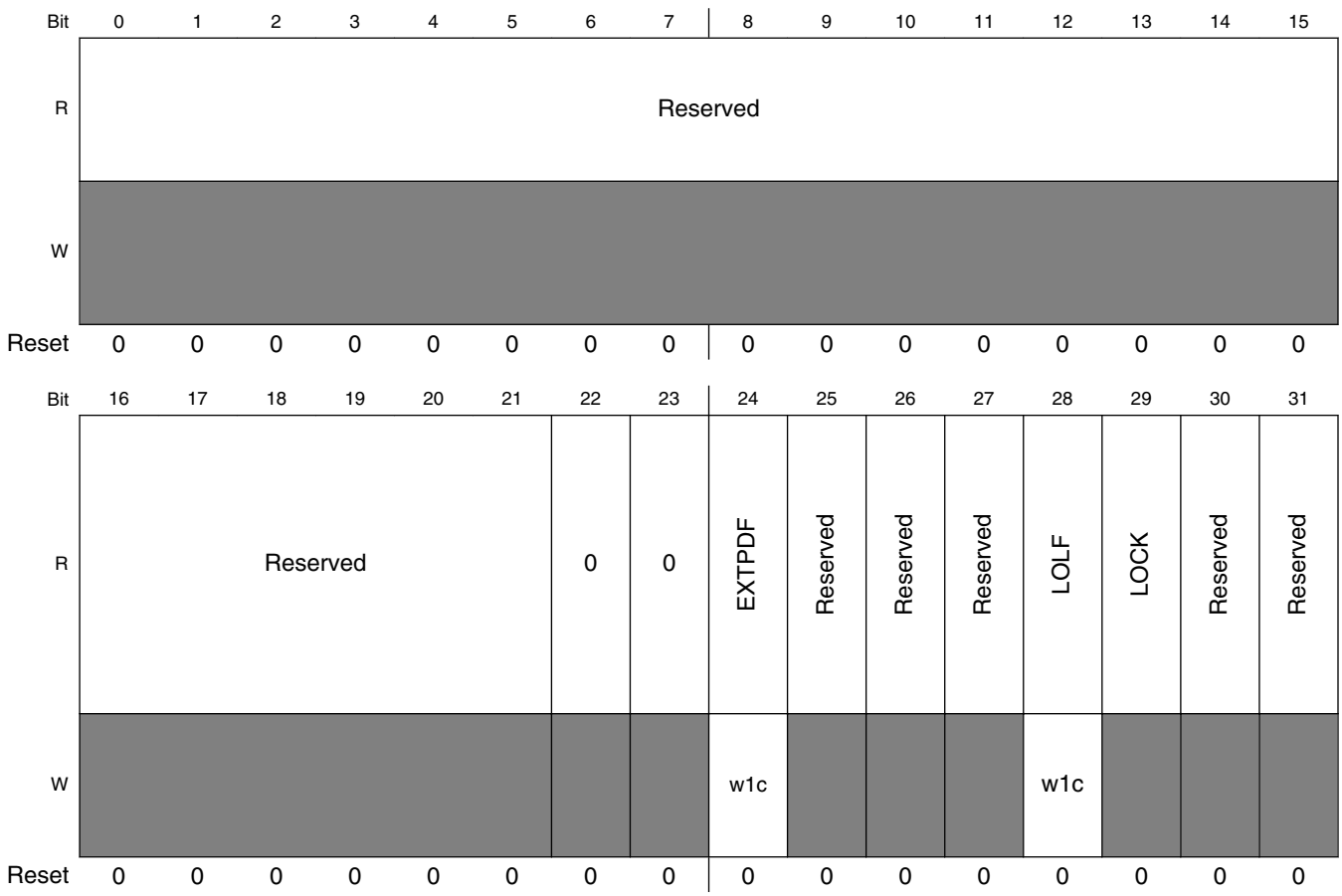
PLLDIG_PLL1CR field descriptions (continued)

Field	Description
	NOTE: CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable. This bit enables generation of an interrupt when external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

27.5.5 PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)

PLL1SR contains status flags showing the current state of the PLL.

Address: 0h base + 24h offset = 24h



PLLDIG_PLL1SR field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested

Table continues on the next page...

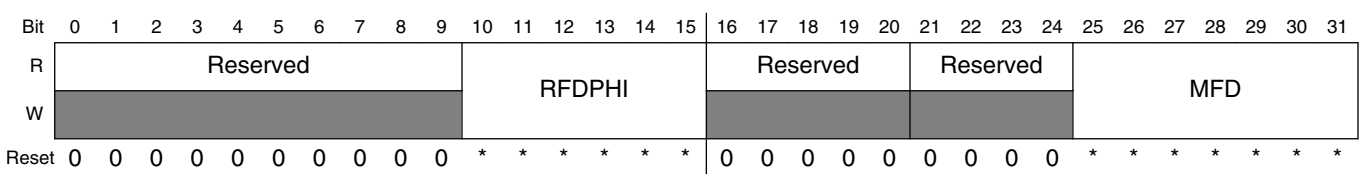
PLLDIG_PLL1SR field descriptions (continued)

Field	Description
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

27.5.6 PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)

The values of the reduced frequency divider (RFDPHI) and loop multiplication factor divider (MFD) can be modified at anytime, but the new values only become effective after the PLL is disabled, then reenabled.

Address: 0h base + 28h offset = 28h



* Notes:
- MFD field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

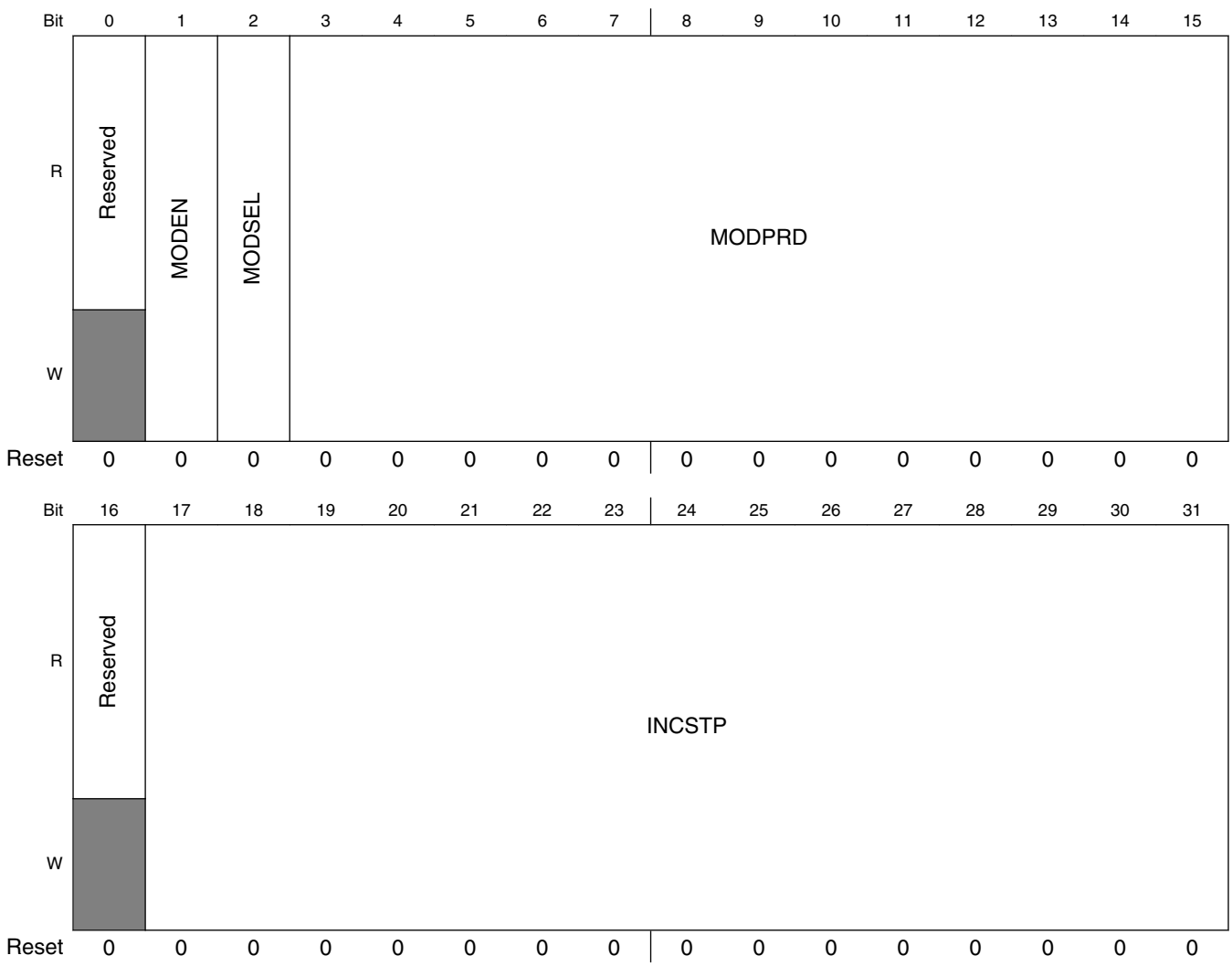
PLLDIG_PLL1DV field descriptions

Field	Description
0–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16–20 Reserved	This field is reserved.
21–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 16...34 (10h...22h). All other values are reserved.

27.5.7 PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)

The PLL1FM register controls enables frequency modulation, and provides controls for modulation mode selection, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). The PLL would then be required to be power cycled to regain normal functionality. Modulation period (PLL1FM[MODPRD]) and increment step fields (PLL1FM[INCSTP]) can be changed without losing lock.

Address: 0h base + 2Ch offset = 2Ch



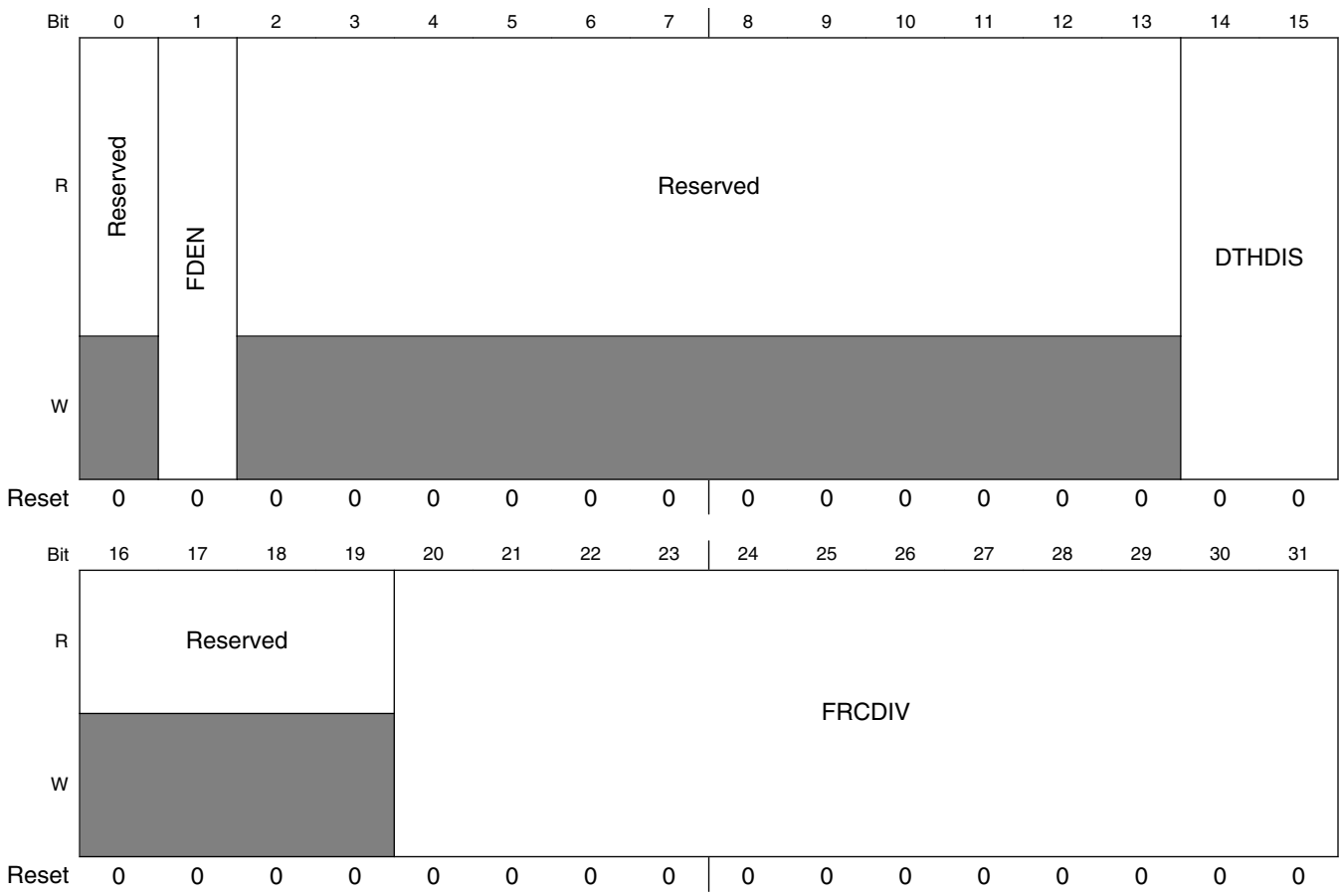
PLLDIG_PLL1FM field descriptions

Field	Description
0 Reserved	This field is reserved.
1 MODEN	Modulation enable. This bit enables the frequency modulation. 0 Frequency modulation disabled. 1 Frequency modulation enabled.
2 MODSEL	Modulation selection. This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Centre Spread Modulation - Modulation centred around nominal frequency. 1 Down Spread Modulation - Modulation spread below nominal frequency.
3–15 MODPRD	Modulation period. MODPRD is the modulation period variable derived from the formulas shown in Frequency modulation.
16 Reserved	This field is reserved.
17–31 INCSTP	Increment step. This field is the INCSTP variable derived from the formulas shown in Frequency modulation.

27.5.8 PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (PLL*n*CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality. The dither disable (PLLDIG_PLL1FD[DTHDIS]) and fractional divider (PLLDIG_PLL1FD[FRCDIV]) fields can be changed without losing lock, however, the output clock from PLL can have an incorrect frequency for a few cycles after either of these updates.

Address: 0h base + 30h offset = 30h



PLLDIG_PLL1FD field descriptions

Field	Description
0 Reserved	This field is reserved.

Table continues on the next page...

PLLDIG_PLL1FD field descriptions (continued)

Field	Description
1 FDEN	Fractional Divide Enable This bit enables the fractional divider in the loop divider for PLL1. 0 Fractional divide disabled. 1 Fractional divide enabled.
2–13 Reserved	This field is reserved.
14–15 DTHDIS	Dither Disable. **Important:** This field must be written 00b. 00 Dither enabled 01 Dither disabled 10 Dither enabled 11 Dither disabled
16–19 Reserved	This field is reserved.
20–31 FRCDIV	Fractional divide input. When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by the equation in the section Clock configuration.

27.6 Register classification for safety requirements

This module is classified as ViMo (Vital Modules) for functional safety. Hence, the registers of this module have been classified as shown in the Functional Safety chapter.

27.7 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

27.7.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for the PLLs are discussed in detail in the *Data Sheet*.

27.7.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV, and PLL1FD registers, and calculated according to the following equations:

$$f_{pll0\_phi} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]} \right)$$

Equation 1. PLL0 PHI Output Frequency

$$f_{pll0\_phi1} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]} \right)$$

Equation 2. PLL0 PHI1 Output Frequency

$$f_{pll1\_phi} = f_{pll1\_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 3. PLL1 PHI Output Frequency

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLLs is determined by the configuration of the PLL1DV, PLL1FD, and PLL0DV registers, according to the following equations:

$$f_{pll0\_VCO} = \frac{f_{pll0\_ref} \times PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}$$

Equation 4. PLL0 VCO Frequency

$$f_{pll1\_VCO} = f_{pll1\_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} \right)$$

Equation 5. PLL1 VCO Frequency

NOTE

f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of the PLL*n*DV registers are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note

The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_<*mode*>_MC before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the input clocks are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in Initialization information.

27.7.3 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 27-1. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

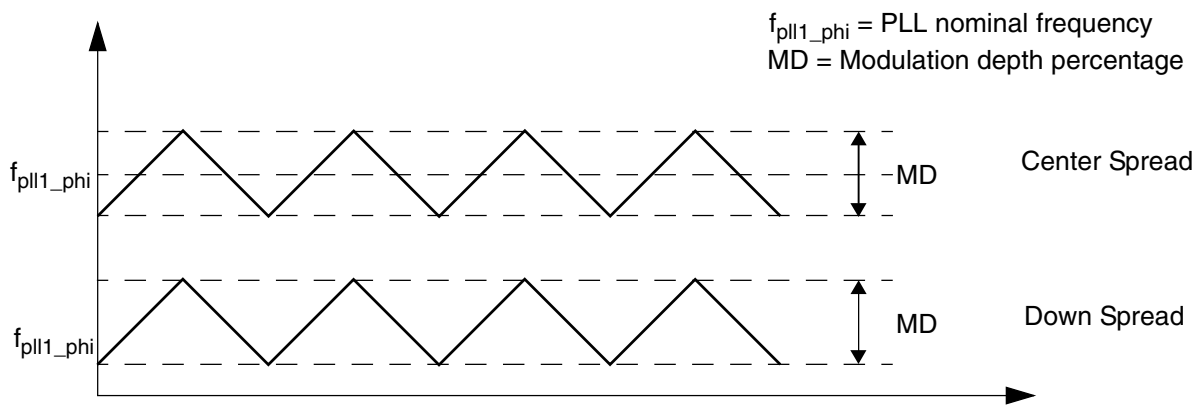


Figure 27-1. Triangular frequency modulation

NOTE

The device maximum operating frequency includes the frequency modulation. If center modulation is used, the f_{sys} must be below f_{max} by MD percentage such that $f_{max} = f_{sys} \times (1 + (MD\% / 100))$.

The following equations define how to calculate PLL1FM[MODPRD] and PLL1FM[INCSTP] based on the output frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD).

$$PLL1FM[MODPRD] = round\left(\frac{f_{pll1\_ref}}{4 \times f_{mod}}\right)$$

Equation 6. PLL1FM[MODPRD] Calculation

The equation to determine PLL1FM[INCSTP] is shown in Equation 7 on page 1131.

$$PLL1FM[INCSTP] = round\left(\frac{(2^{15}-1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

Equation 7. PLL1FM[INCSTP] Calculation

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Equation 8. PLL1FM[MODPRD] and PLL1FM[INCSTP] Restriction

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is shown in Equation 9 on page 1131.

$$ModulationDepth = round\left(\frac{PLL1FM[MODPRD] \times PLL1FM[INCSTP] \times 100 \times 5}{(2^{15} - 1) \times PLL1DV[MFD]}\right)$$

Equation 9. Modulation Depth

FM parameters should only be changed, and FM enabled, after PLL0 has obtained lock. The sequence for reprogramming the FM is:

1. Power down PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 0, followed by a mode change.

2. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].
3. Program the PLL1FM while PLL1 is powered down.
4. Power up the PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 1, followed by a mode change.
5. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].

27.7.4 Maximum lock time

This describes the maximum lock time of the PLL across process, supply voltages, and junction temperature. Maximum lock time is for the condition when the operating mode of the PLL has changed from power-down to Normal mode, input control bits are stable, and supplies (analog and digital) have attained levels shown in the *Data Sheet*. It is also the time in which the PLLs regain lock after a loss of lock event, assuming all inputs are stable and supplies are within specifications.

27.8 Initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The Dual PLLDIG initialization procedure is described in the "Clocking" chapter.

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

Chapter 28
Clock Monitor Unit (CMU)

28.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock source CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and detects if the monitored clock frequency leaves an upper or lower frequency boundary

NOTE

See the "Clocking" chapter for chip-specific sources used by the CMU.

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take the necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate an event if CLKMN1 is greater than a high frequency boundary or less than a low frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of a clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

NOTE

See the "Clocking" chapter for the number of CMU instances on this chip.

28.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ n clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN ÷ 4.
- Event generation for various failures detected inside monitoring unit.

28.2 Block diagram

The block diagram of the CMU module(s) is shown in the “Clocking" chapter of this Reference Manual.

28.3 Signals

The table below describes the signals on the boundary of the CMU (in alphabetical order).

Table 28-1. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

NOTE

See the "Clocking" chapter for device specific clock sources of each CMU.

28.4 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in the following table.

NOTE

See "Clocking" chapter for register and field availability details.

CMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMU Control Status Register (CMU_CSR)	32	R/W	See section	28.4.1/1136
4	CMU Frequency Display Register (CMU_FDR)	32	R	0000_0000h	28.4.2/1137
8	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	32	R/W	0000_0FFFh	28.4.3/1138
C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	32	R/W	0000_0000h	28.4.4/1138
10	CMU Interrupt Status Register (CMU_ISR)	32	w1c	See section	28.4.5/1139
18	CMU Measurement Duration Register (CMU_MDR)	32	R/W	0000_0000h	28.4.6/1140

28.4.1 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SFM	0						
W																
Reset	0	0	0	0	0	0	0	0	0*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CKSEL1		0					RCDIV		CME
W																
Reset	0	0	0	0	0	0	0*	0*	0	0	0	0	0	0*	0*	0

* Notes:
- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 RCDIV	CLKMT0_RMN division factor.

Table continues on the next page...

CMU_CSR field descriptions (continued)

Field	Description
	These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{CLKMT0\_RMN} \div 2^{RCDIV}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. 00 CLKMT0_RMN ÷ 1 (No division) 01 CLKMT0_RMN ÷ 2 10 CLKMT0_RMN ÷ 4 11 CLKMT0_RMN ÷ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

28.4.2 CMU Frequency Display Register (CMU_FDR)

The FDR is used to determine the measured frequency being monitored by the CMU.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

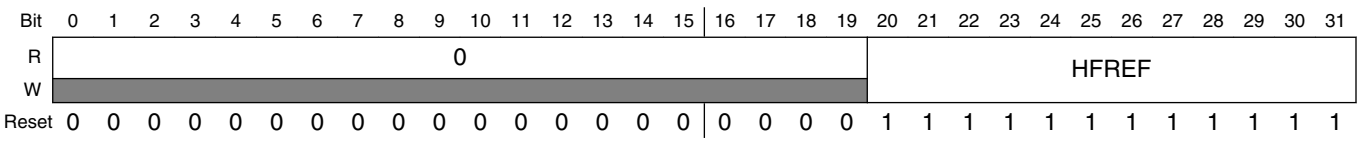
CMU_FDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0\_RMT} \times CMU\_MDR[MD]) \div CMU\_FDR[FD]$

28.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the HFREFR register.

Address: 0h base + 8h offset = 8h



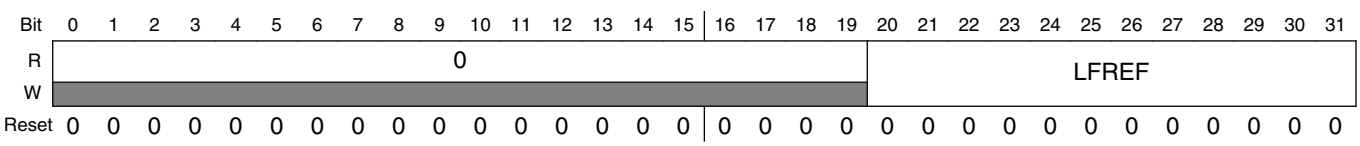
CMU_HFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HFREF	High Frequency reference value. These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

28.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the LFREFR register.

Address: 0h base + Ch offset = Ch



CMU_LFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMU_LFREFR field descriptions (continued)

Field	Description
20–31 LFREF	Low Frequency reference value. These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

28.4.5 CMU Interrupt Status Register (CMU_ISR)

NOTE

All the flags in CMU_ISR set asynchronously. This register must be read only after an interrupt(safe mode entry) is triggered by the CMUas confirmed by MC_RGM. Otherwise, the read access on this register may fetch an incorrect value.

NOTE

Before entering low-power stop modes, all clock frequency measurements in CMU should be disabled. Failing to do so may result in spurious interrupts.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												Reserved	FHHI	FLLI	OLRI
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*

* Notes:
- OLRI field: Not all CMU blocks will utilize this feature. See the "Clocking" chapter for specific CMU implementation details.

CMU_ISR field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 FHHI	CLKMN1 frequency higher than high reference event status. This bit is set by hardware when CLKMN1 frequency becomes higher than HFREF value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status. This bit is set by hardware when CLKMN1 frequency becomes lower than LFREF value, and CLKMN1 is 'ON' as signaled by the MC_ME. Software clears this field by writing a '1'. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{CLKMT0_RMN} \div 2^{RCDIV}$event status. This bit is set by hardware when the f_{CLKMN0_RMT} is less than $f_{CLKMT0_RMN} \div 2^{RCDIV}$frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. **NOTE:** While entering STOP mode, OLRI may be triggered. To avoid this when the system is running on the PLL, disable XOSC before entering STOP mode. 0 No OLR event 1 OLR event occurred

28.4.6 CMU Measurement Duration Register (CMU_MDR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																MD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_MDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 MD	Measurement duration bits

Table continues on the next page...

CMU_MDR field descriptions (continued)

Field	Description
	This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

28.5 Functional description

28.5.1 Frequency meter

The purpose of the frequency meter is to evaluate the deviation from the nominal metered source (such as CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of the metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the "Clocking" chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR as shown in the following equation:

$$f_{sel} = f_{CLKMN0\_RMT} \times \frac{CMU\_MDR[MD]}{CMU\_FDR[FD]}$$

28.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than the frequency of CLKMT0_RMN $\div 2^{CMU\_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note

f_{CLKMN0_RMT} must be greater than $f_{CLKMT0\_RMN} \div 2^{CMU\_CSR[RCDIV]}$ by at least 0.5 MHz in order to guarantee correct f_{CLKMN0_RMT} monitoring.

28.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note

An example of determining the $HFREF_{Actual}$ is as follows. Assume a f_{CLKMT0_RMN} = 16 MHz with a accuracy of +/-5%. In order to monitor a f_{CLKMN1} = 200 MHz, the ideal $HFREF_{Ideal}$ = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration ($HFREF_{Actual} = (HFREF_{Ideal} \div 0.95)$.

The actual LFREF value will be 762 when accuracy is taken into consideration ($LFREF_{Actual} = LFREF_{Ideal} \div 1.05$).

Chapter 29
Clock Generation Module (MC_CGM)

29.1 Introduction

29.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

Figure 29-1 shows the MC_CGM block diagram.

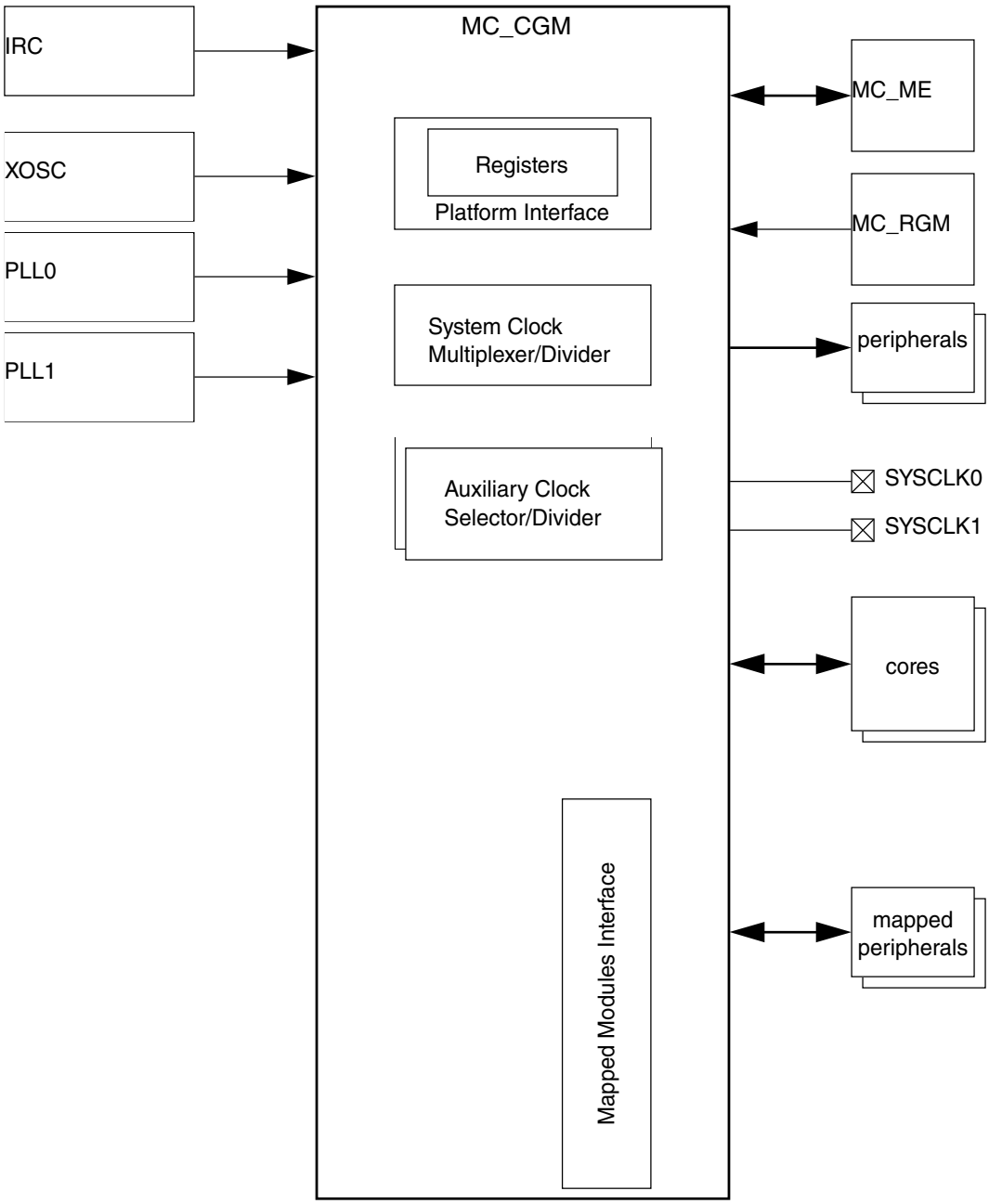


Figure 29-1. MC_CGMBlock Diagram

29.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control

- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- supports multiple clock sources and maps their address spaces to its memory map
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

29.2 External signal description

The MC_CGM delivers output clocks to the SYSCLK0 and SYSCLK1 pins for off-chip use and/or observation.

29.3 Memory map and register definition

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0704, as a half-word at address offset 0x0706, or as a byte at address offset 0x0707.

NOTE

All registers will be accessible in "User mode" provided the access to this mode is enabled through REGPROT settings. Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error(only if SSCM_ERROR[RAE] is set)

MC_CGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
680	Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (MC_CGM_AC5_CDC0)	32	R/W	0000_0000h	29.3.1/1148
684	Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (MC_CGM_AC5_CDC1)	32	R/W	0000_0000h	29.3.2/1149

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
688	Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (MC_CGM_AC5_CDC2)	32	R/W	0000_0000h	29.3.3/1150
690	Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (MC_CGM_AC5_CDC10)	32	R/W	0000_0000h	29.3.4/1151
694	Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (MC_CGM_AC5_CDC11)	32	R/W	0000_0000h	29.3.5/1152
698	Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (MC_CGM_AC5_CDC12)	32	R/W	0000_0000h	29.3.6/1152
69C	Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (MC_CGM_AC5_CDC13)	32	R/W	0000_0000h	29.3.7/1153
6A0	Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (MC_CGM_AC5_CDC20)	32	R/W	0000_0000h	29.3.8/1154
6A4	Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (MC_CGM_AC5_CDC21)	32	R/W	0000_0000h	29.3.9/1155
6A8	Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (MC_CGM_AC5_CDC22)	32	R/W	0000_0000h	29.3.10/ 1155
6AC	Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (MC_CGM_AC5_CDC23)	32	R/W	0000_0000h	29.3.11/ 1156
700	PCS Switch Duration Register (MC_CGM_PCS_SDUR)	8	R/W	00h	29.3.12/ 1157
704	PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)	32	R/W	03E7_0000h	29.3.13/ 1157
708	PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)	32	R/W	0000_03E7h	29.3.14/ 1158
70C	PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)	32	R/W	0000_03E7h	29.3.15/ 1159
710	PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)	32	R/W	03E7_0000h	29.3.16/ 1159
714	PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)	32	R/W	0000_03E7h	29.3.17/ 1160
718	PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)	32	R/W	0000_03E7h	29.3.18/ 1160
728	PCS Divider Change Register 4 (MC_CGM_PCS_DIVC4)	32	R/W	03E7_0000h	29.3.19/ 1161
72C	PCS Divider End Register 4 (MC_CGM_PCS_DIVE4)	32	R/W	0000_03E7h	29.3.20/ 1162
730	PCS Divider Start Register 4 (MC_CGM_PCS_DIVS4)	32	R/W	0000_03E7h	29.3.21/ 1162
7D0	System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)	32	R/W	0000_0001h	29.3.22/ 1163
7D4	Divider Update Type Register (MC_CGM_DIV_UPD_TYPE)	32	R/W	0000_0000h	29.3.23/ 1164
7D8	Divider Update Trigger Register (MC_CGM_DIV_UPD_TRIG)	32	W (always reads 0)	0000_0000h	29.3.24/ 1165

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7DC	Divider Update Status Register (MC_CGM_DIV_UPD_STAT)	32	R/W	0000_0000h	29.3.25/ 1166
7E4	System Clock Select Status Register (MC_CGM_SC_SS)	32	R	0008_0000h	29.3.26/ 1167
7E8	System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)	32	R/W	8000_0000h	29.3.27/ 1168
7EC	System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)	32	R/W	8000_0000h	29.3.28/ 1169
7F0	System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)	32	R/W	8000_0000h	29.3.29/ 1170
7F4	System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)	32	R/W	8000_0000h	29.3.30/ 1170
7F8	System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)	32	R/W	8000_0000h	29.3.31/ 1171
800	Auxiliary Clock 0 Select Control Register (MC_CGM_AC0_SC)	32	R/W	0100_0000h	29.3.32/ 1172
804	Auxiliary Clock 0 Select Status Register (MC_CGM_AC0_SS)	32	R	0100_0000h	29.3.33/ 1173
808	Auxiliary Clock 0 Divider 0 Configuration Register (MC_CGM_AC0_DC0)	32	R/W	8000_0000h	29.3.34/ 1174
80C	Auxiliary Clock 0 Divider 1 Configuration Register (MC_CGM_AC0_DC1)	32	R/W	0000_0000h	29.3.35/ 1175
810	Auxiliary Clock 0 Divider 2 Configuration Register (MC_CGM_AC0_DC2)	32	R/W	0000_0000h	29.3.36/ 1175
814	Auxiliary Clock 0 Divider 3 Configuration Register (MC_CGM_AC0_DC3)	32	R/W	0000_0000h	29.3.37/ 1176
818	Auxiliary Clock 0 Divider 4 Configuration Register (MC_CGM_AC0_DC4)	32	R/W	0000_0000h	29.3.38/ 1177
820	Auxiliary Clock 1 Select Control Register (MC_CGM_AC1_SC)	32	R/W	0100_0000h	29.3.39/ 1178
824	Auxiliary Clock 1 Select Status Register (MC_CGM_AC1_SS)	32	R	0100_0000h	29.3.40/ 1179
828	Auxiliary Clock 1 Divider 0 Configuration Register (MC_CGM_AC1_DC0)	32	R/W	0000_0000h	29.3.41/ 1180
848	Auxiliary Clock 2 Divider 0 Configuration Register (MC_CGM_AC2_DC0)	32	R/W	0000_0000h	29.3.42/ 1181
84C	Auxiliary Clock 2 Divider 1 Configuration Register (MC_CGM_AC2_DC1)	32	R/W	0000_0000h	29.3.43/ 1181
860	Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)	32	R/W	0000_0000h	29.3.44/ 1182
864	Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)	32	R	0000_0000h	29.3.45/ 1183
880	Auxiliary Clock 4 Select Control Register (MC_CGM_AC4_SC)	32	R/W	0100_0000h	29.3.46/ 1184

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
884	Auxiliary Clock 4 Select Status Register (MC_CGM_AC4_SS)	32	R	0100_0000h	29.3.47/ 1185
8A8	Auxiliary Clock 5 Divider 0 Configuration Register (MC_CGM_AC5_DC0)	32	R/W	0000_0000h	29.3.48/ 1186
8AC	Auxiliary Clock 5 Divider 1 Configuration Register (MC_CGM_AC5_DC1)	32	R/W	0000_0000h	29.3.49/ 1187
8B0	Auxiliary Clock 5 Divider 2 Configuration Register (MC_CGM_AC5_DC2)	32	R/W	0000_0000h	29.3.50/ 1187
8C0	Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)	32	R/W	0000_0000h	29.3.51/ 1188
8C4	Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)	32	R	0000_0000h	29.3.52/ 1189
8C8	Auxiliary Clock 6 Divider 0 Configuration Register (MC_CGM_AC6_DC0)	32	R/W	0000_0000h	29.3.53/ 1190
8E0	Auxiliary Clock 7 Select Control Register (MC_CGM_AC7_SC)	32	R/W	0000_0000h	29.3.54/ 1191
8E4	Auxiliary Clock 7 Select Status Register (MC_CGM_AC7_SS)	32	R	0000_0000h	29.3.55/ 1192
8E8	Auxiliary Clock 7 Divider 0 Configuration Register (MC_CGM_AC7_DC0)	32	R/W	0000_0000h	29.3.56/ 1193
900	Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)	32	R/W	0100_0000h	29.3.57/ 1193
904	Auxiliary Clock 8 Select Status Register (MC_CGM_AC8_SS)	32	R	0100_0000h	29.3.58/ 1194
908	Auxiliary Clock 8 Divider 0 Configuration Register (MC_CGM_AC8_DC0)	32	R/W	0000_0000h	29.3.59/ 1195
920	Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)	32	R/W	0000_0000h	29.3.60/ 1196
924	Auxiliary Clock 9 Select Status Register (MC_CGM_AC9_SS)	32	R	0000_0000h	29.3.61/ 1197
928	Auxiliary Clock 9 Divider 0 Configuration Register (MC_CGM_AC9_DC0)	32	R/W	0000_0000h	29.3.62/ 1198
940	Auxiliary Clock 10 Select Control Register (MC_CGM_AC10_SC)	32	R/W	0100_0000h	29.3.63/ 1198
944	Auxiliary Clock 10 Select Status Register (MC_CGM_AC10_SS)	32	R	0100_0000h	29.3.64/ 1199
948	Auxiliary Clock 10 Divider 0 Configuration Register (MC_CGM_AC10_DC0)	32	R/W	0000_0000h	29.3.65/ 1200

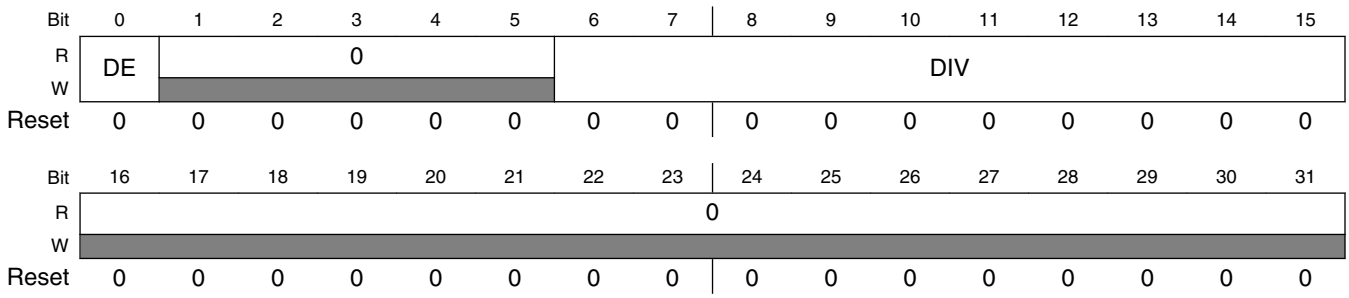
29.3.1 Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (MC_CGM_AC5_CDC0)

This register controls auxiliary clock 5 cascaded divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 680h offset = 680h



MC_CGM_AC5_CDC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 0 1 Enable auxiliary clock 5 cascaded divider 0
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

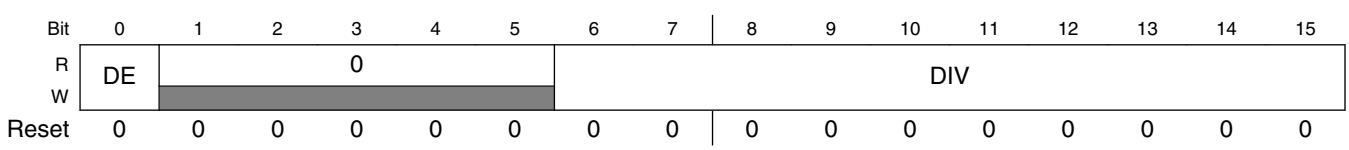
29.3.2 Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (MC_CGM_AC5_CDC1)

This register controls system clock divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 684h offset = 684h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 1 1 Enable auxiliary clock 5 cascaded divider 1
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.3 Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (MC_CGM_AC5_CDC2)

This register controls system clock divider 2.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 688h offset = 688h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0					DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC2 field descriptions

Field	Description
0 DE	Divider Enable

Table continues on the next page...

MC_CGM_AC5_CDC2 field descriptions (continued)

Field	Description
	0 Disable auxiliary clock 5 cascaded divider 2 1 Enable auxiliary clock 5 cascaded divider 2
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.4 Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (MC_CGM_AC5_CDC10)

This register controls auxiliary clock 5 cascaded divider 10.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 690h offset = 690h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0					DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC10 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 10 1 Enable auxiliary clock 5 cascaded divider 10
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC1[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.5 Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (MC_CGM_AC5_CDC11)

This register controls auxiliary clock 5 cascaded divider 11.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 694h offset = 694h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC11 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 11 1 Enable auxiliary clock 5 cascaded divider 11
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC1[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

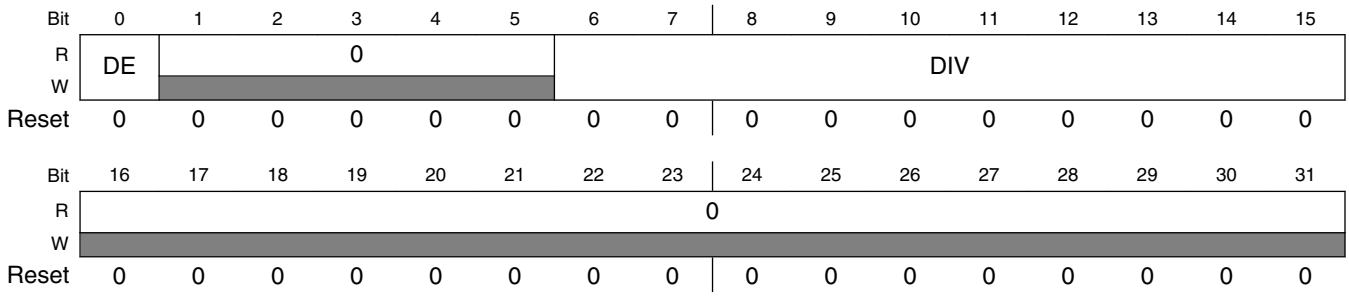
29.3.6 Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (MC_CGM_AC5_CDC12)

This register controls auxiliary clock 5 cascaded divider 12.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 698h offset = 698h



MC_CGM_AC5_CDC12 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 12 1 Enable auxiliary clock 5 cascaded divider 12
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC1[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

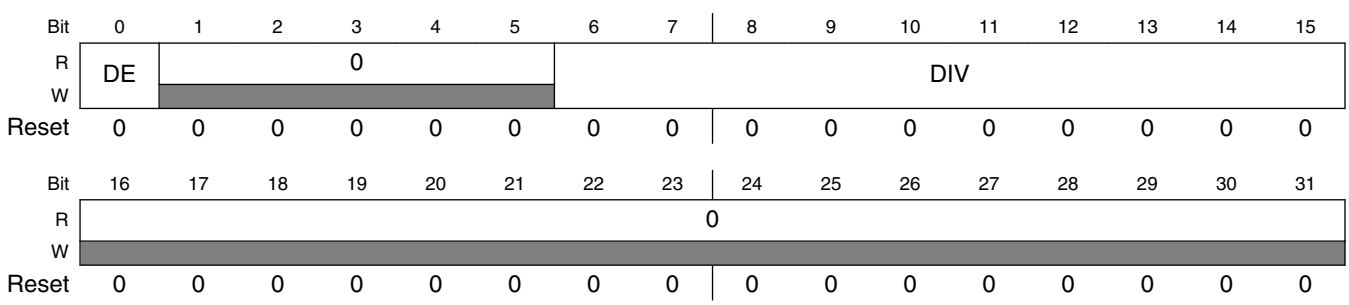
29.3.7 Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (MC_CGM_AC5_CDC13)

This register controls auxiliary clock 5 cascaded divider 13.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 69Ch offset = 69Ch



MC_CGM_AC5_CDC13 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 13 1 Enable auxiliary clock 5 cascaded divider 13
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC1[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.8 Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (MC_CGM_AC5_CDC20)

This register controls auxiliary clock 5 cascaded divider 20.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 6A0h offset = 6A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC20 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 20 1 Enable auxiliary clock 5 cascaded divider 20
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC2[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

Table continues on the next page...

MC_CGM_AC5_CDC20 field descriptions (continued)

Field	Description
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.9 Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (MC_CGM_AC5_CDC21)

This register controls auxiliary clock 5 cascaded divider 21.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 6A4h offset = 6A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC21 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 21 1 Enable auxiliary clock 5 cascaded divider 21
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC2[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

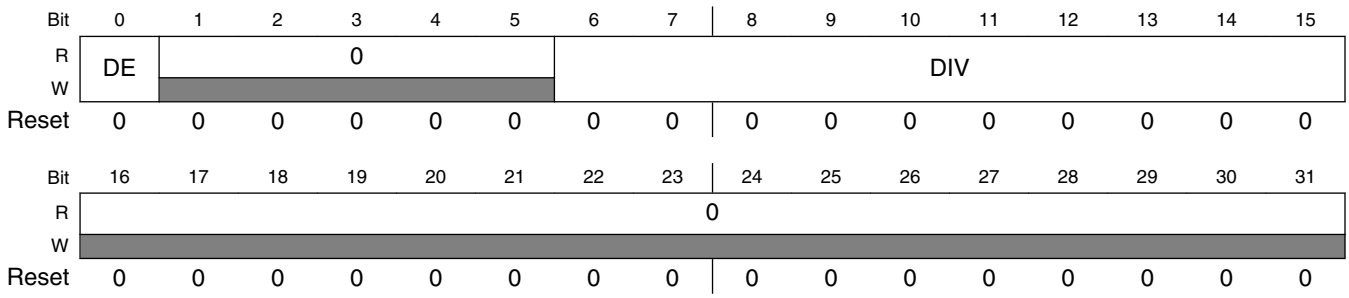
29.3.10 Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (MC_CGM_AC5_CDC22)

This register controls auxiliary clock 5 cascaded divider 22.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 6A8h offset = 6A8h



MC_CGM_AC5_CDC22 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 22 1 Enable auxiliary clock 5 cascaded divider 22
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC2[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

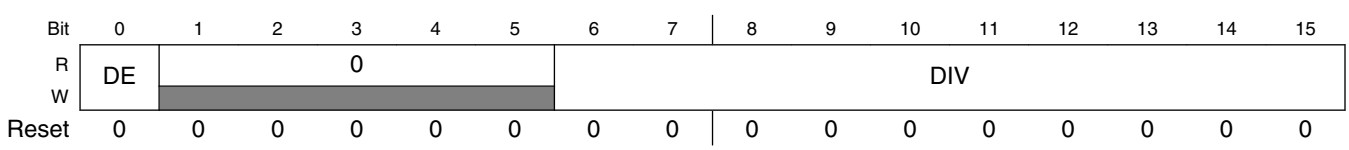
29.3.11 Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (MC_CGM_AC5_CDC23)

This register controls auxiliary clock 5 cascaded divider 23.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 6ACh offset = 6ACh



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_CDC23 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 23 1 Enable auxiliary clock 5 cascaded divider 23
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant peripheral clock will have a period '(DIV + 1) x (CGM_AC5_CDC2[DIV] + 1) x (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.12 PCS Switch Duration Register (MC_CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See Progressive system clock switching for details on how to set this value.

Address: 0h base + 700h offset = 700h

Bit	0	1	2	3	4	5	6	7
Read Write	SDUR							
Reset	0	0	0	0	0	0	0	0

MC_CGM_PCS_SDUR field descriptions

Field	Description
0–7 SDUR	Switch Duration - This value defines the duration of one PCS clock switch step in terms of 16 MHz int. RC osc. cycles.

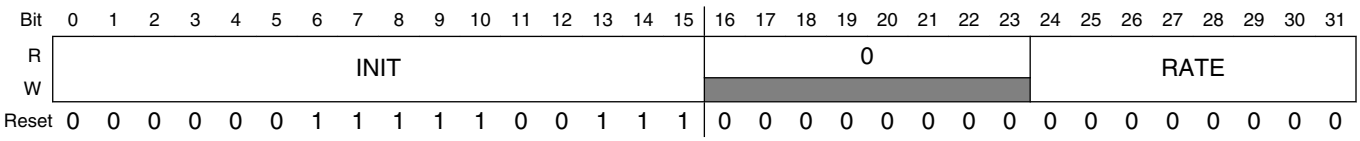
29.3.13 PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the external crystal osc. on ramp-up and ramp-down, respectively. See Progressive system clock switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 704h offset = 704h



MC_CGM_PCS_DIVC1 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to the external crystal osc..
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the external crystal osc..

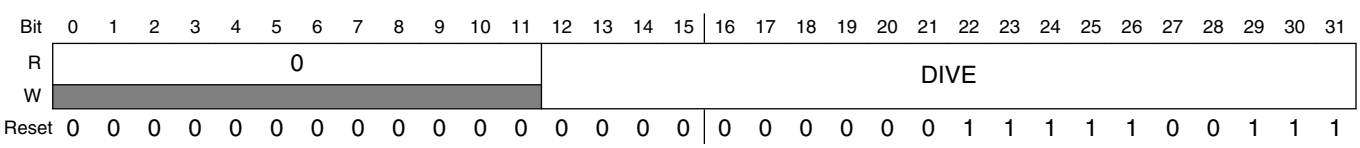
29.3.14 PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the external crystal osc. on ramp-down. See Progressive system clock switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 708h offset = 708h



MC_CGM_PCS_DIVE1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the external crystal osc..

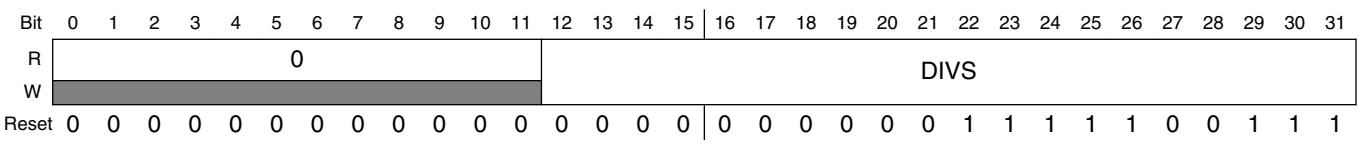
29.3.15 PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the external crystal osc. on ramp-up. Register *n* corresponds to system clock source *n* . See Progressive system clock switching for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 70Ch offset = 70Ch



MC_CGM_PCS_DIVS1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the external crystal osc..

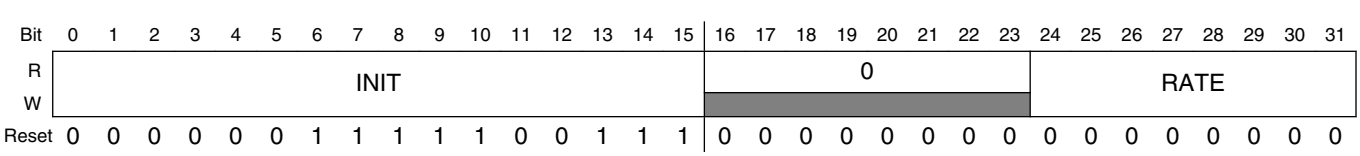
29.3.16 PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL0 PHI on ramp-up and ramp-down, respectively. See Progressive system clock switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 710h offset = 710h



MC_CGM_PCS_DIVC2 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL0 PHI.

29.3.17 PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0 PHI on ramp-down. See Progressive system clock switching for details on how to set This value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 714h offset = 714h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVE2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

29.3.18 PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0 PHI on ramp-up. See Progressive system clock switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 718h offset = 718h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVS2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.

29.3.19 PCS Divider Change Register 4 (MC_CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL1 PHI on ramp-up and ramp-down, respectively. See Progressive system clock switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 728h offset = 728h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_PCS_DIVC4 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL1 PHI.

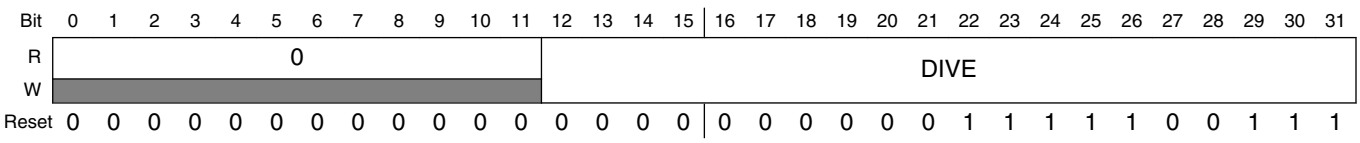
29.3.20 PCS Divider End Register 4 (MC_CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL1 PHI on ramp-down. See Progressive system clock switching for details on how to set This value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 72Ch offset = 72Ch



MC_CGM_PCS_DIVE4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL1 PHI.

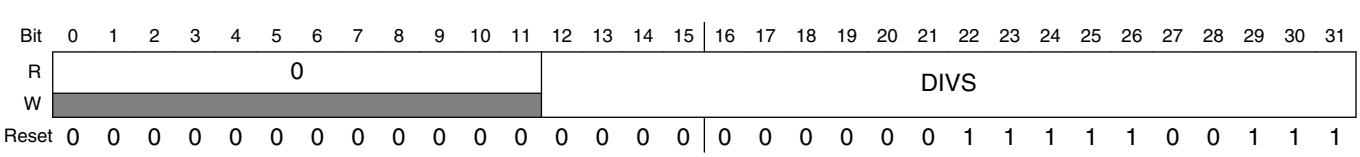
29.3.21 PCS Divider Start Register 4 (MC_CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL1 PHI on ramp-up. See Progressive system clock switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 730h offset = 730h



MC_CGM_PCS_DIVS4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.

29.3.22 System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

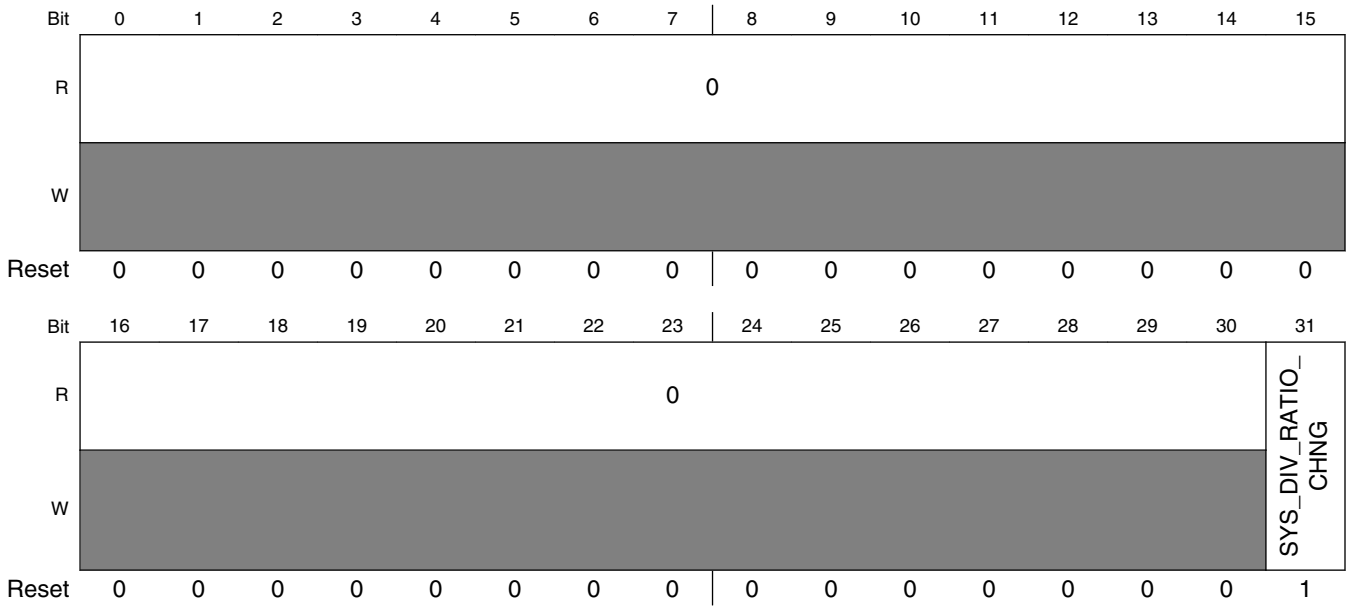
An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations. An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3. The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

See Divider update triggering for details.

CAUTION

Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

Address: 0h base + 7D0h offset = 7D0h



MC_CGM_SC_DIV_RC field descriptions

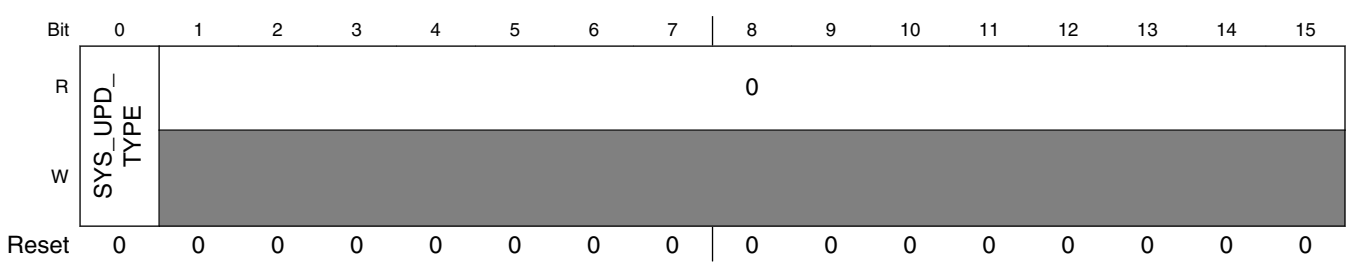
Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SYS_DIV_ RATIO_CHNG	System Clock Divider Ratio Change 0 The system clock divider ratios will not change with the next system clock divider configuration update 1 The system clock divider ratios will change with the next system clock divider configuration update

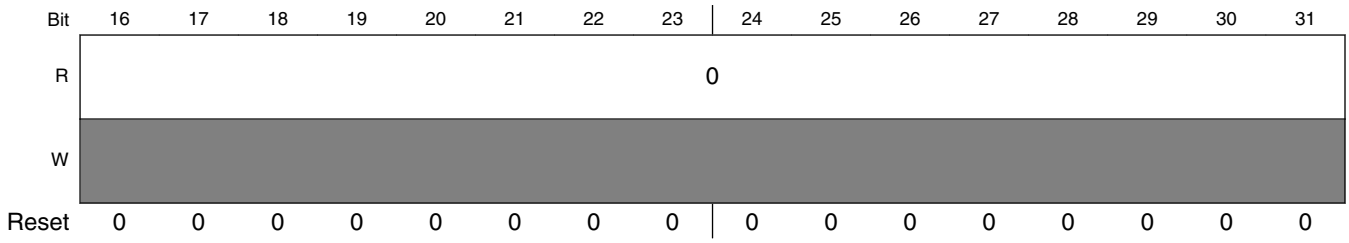
29.3.23 Divider Update Type Register (MC_CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers will take effect.

See Divider update triggering for details.

Address: 0h base + 7D4h offset = 7D4h





MC_CGM_DIV_UPD_TYPE field descriptions

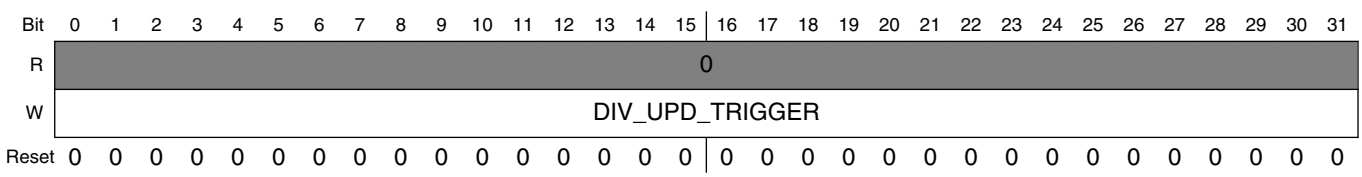
Field	Description
0 SYS_UPD_TYPE	System Clock Divider Update Type 0 The configuration for each system clock divider is updated immediately on writing to the corresponding CGM_SC_DCi register 1 The configuration for each system clock divider is pre-loaded on writing to the corresponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.24 Divider Update Trigger Register (MC_CGM_DIV_UPD_TRIG)

Writing any value to this register will cause all dividers that have corresponding bits set to ‘1’ in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

See Divider update triggering for details.

Address: 0h base + 7D8h offset = 7D8h



MC_CGM_DIV_UPD_TRIG field descriptions

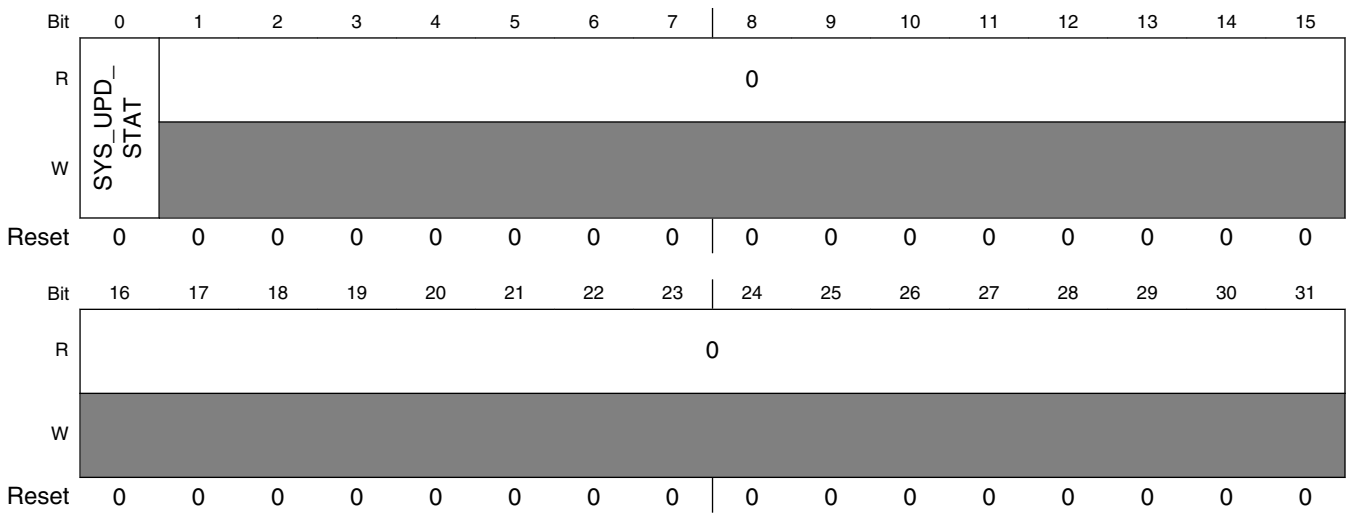
Field	Description
0–31 DIV_UPD_ TRIGGER	Writing any value to this register will cause all dividers that have corresponding bits set to ‘1’ in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

29.3.25 Divider Update Status Register (MC_CGM_DIV_UPD_STAT)

This register indicates whether a divider for each clock is in the process of being updated. If the CGM_DIV_UPD_TYPE[i] is ‘0’, CGM_DIV_UPD_STAT[i] is set to ‘1’ immediately on writing to one of the corresponding divider configuration registers. Otherwise, CGM_DIV_UPD_STAT[i] is set to ‘1’ immediately on writing to the CGM_DIV_UPD_TRIG register if one of the corresponding divider configurations has been pre-loaded. CGM_DIV_UPD_STAT[i] is cleared to ‘0’ as soon as all the corresponding dividers have been updated and the dividers generate the desired frequency.

See Divider update triggering for details.

Address: 0h base + 7DCh offset = 7DCh



MC_CGM_DIV_UPD_STAT field descriptions

Field	Description
0 SYS_UPD_STAT	System Clock Divider Update Status 0 The configuration for at least one system clock divider is not being updated 1 The configuration for at least one system clock divider is being updated
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.26 System Clock Select Status Register (MC_CGM_SC_SS)

This register provides the current system clock source selection.

Address: 0h base + 7E4h offset = 7E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				SELSTAT				0				SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	System Clock Source Selection Status - This value indicates the current source for the system clock. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–14 SWTRG	Switch Trigger cause - This value indicates the cause for the latest clock source switch. 000 reserved 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock

Table continues on the next page...

MC_CGM_SC_SS field descriptions (continued)

Field	Description
	011 switch after request from MC_ME failed due inactive current clock 100 switch to 16 MHz int. RC osc. due to SAFE mode request or reset succeeded 101 switch to 16 MHz int. RC osc. due to SAFE mode request or reset succeeded, but current clock source was inactive 110 reserved 111 reserved
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

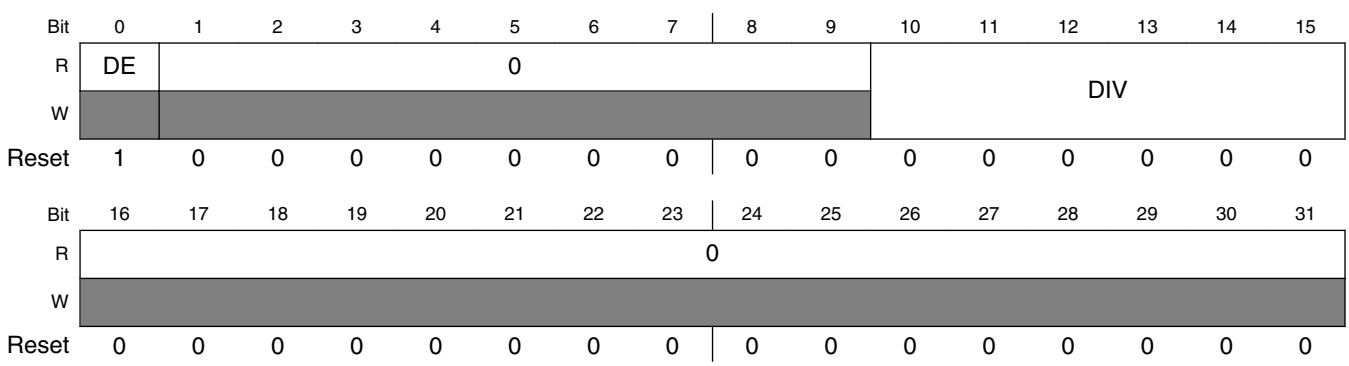
29.3.27 System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)

This register controls system clock divider 0. See System clock dividers for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7E8h offset = 7E8h



MC_CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 1 Enable - This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_SC_DC0 field descriptions (continued)

Field	Description
10–15 DIV	Divider 1 Division Value - The resultant fast XBAR clock will have a period 'DIV + 1' times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.28 System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)

This register controls system clock divider 1. See System clock dividers for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7ECh offset = 7ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC1 field descriptions

Field	Description
0 DE	Divider 1 Enable - This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 1 Division Value - The resultant slow XBAR clock will have a period 'DIV + 1' times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.29 System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)

This register controls system clock divider 2. See Figure 29-5 for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7F0h offset = 7F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC2 field descriptions

Field	Description
0 DE	Divider 1 Enable - This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 2 Division Value - The resultant AIPS clock will have a period 'DIV + 1' times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.30 System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)

This register controls system clock divider 3. See Figure 29-5 for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7F4h offset = 7F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC3 field descriptions

Field	Description
0 DE	Divider 1 Enable - This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 2 Division Value - The resultant COMP/CHKR clock will have a period 'DIV + 1' times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.31 System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)

This register controls system clock divider 4. See Figure 29-5 for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7F8h offset = 7F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC4 field descriptions

Field	Description
0 DE	Divider 1 Enable - This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 2 Division Value - The resultant CLKOUT will have a period 'DIV + 1' times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.32 Auxiliary Clock 0 Select Control Register (MC_CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 0 divider 0: peripheral clock
- divided by auxiliary clock 0 divider 1: sigma delta ADC clock
- divided by auxiliary clock 0 divider 2: SAR ADC clock
- divided by auxiliary clock 0 divider 3: DSPI clock 0
- divided by auxiliary clock 0 divider 4: DSPI clock 1

See Figure 29-5 for details.

Address: 0h base + 800h offset = 800h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 0 Source Selection Control - Selects the source for auxiliary clock 0. 0000 16 MHz int. RC osc. 0001 external crystal osc.

Table continues on the next page...

MC_CGM_AC0_SC field descriptions (continued)

Field	Description
	0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.33 Auxiliary Clock 0 Select Status Register (MC_CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0h base + 804h offset = 804h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 0 Source Selection Status - This value indicates the current source for auxiliary clock 0. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved

Table continues on the next page...

MC_CGM_AC0_SS field descriptions (continued)

Field	Description
	1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.34 Auxiliary Clock 0 Divider 0 Configuration Register (MC_CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 808h offset = 808h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value - The resultant peripheral clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.

Table continues on the next page...

MC_CGM_AC0_DC0 field descriptions (continued)

Field	Description
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.35 Auxiliary Clock 0 Divider 1 Configuration Register (MC_CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 80Ch offset = 80Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0								DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider Division Value - The resultant sigma delta ADC clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the sigma delta ADC clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.36 Auxiliary Clock 0 Divider 2 Configuration Register (MC_CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 810h offset = 810h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider Division Value - The resultant SAR ADC clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the SAR ADC clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.37 Auxiliary Clock 0 Divider 3 Configuration Register (MC_CGM_AC0_DC3)

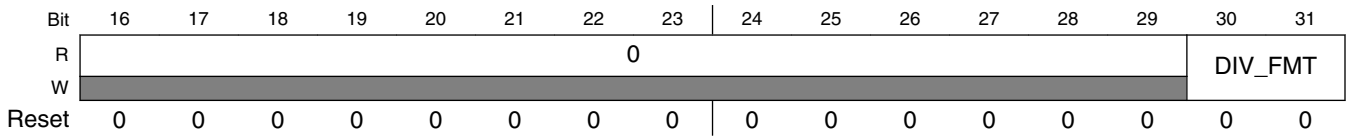
This register controls auxiliary clock 0 divider 3. See Fractional divider details for divider characteristics.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 814h offset = 814h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MC_CGM_AC0_DC3 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 3 1 Enable auxiliary clock 0 divider 3
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–15 DIV	Divider Division Value - The resultant DSPI clock 0 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI clock 0 remains disabled.
16–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 DIV_FMT	Divider Value Format - This failure indicates the format of the division value for calculating the division factor. Note: Choose the DIV_FMT value so that the post Aux_Div register frequency value is not more than the source clock; otherwise, the clock will not be correct. 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

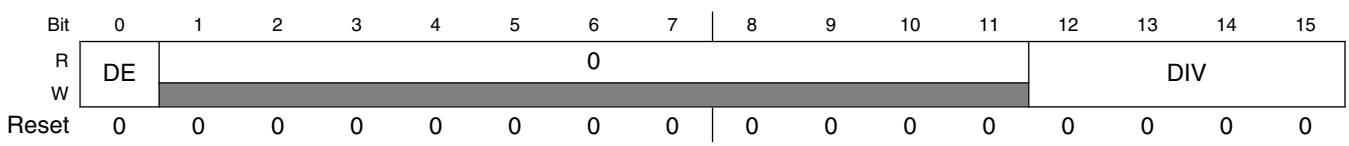
29.3.38 Auxiliary Clock 0 Divider 4 Configuration Register (MC_CGM_AC0_DC4)

This register controls auxiliary clock 0 divider 4.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 818h offset = 818h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC4 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 4 1 Enable auxiliary clock 0 divider 4
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value - The resultant DSPI clock 1 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the SAR ADC clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.39 Auxiliary Clock 1 Select Control Register (MC_CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 1 divider 0: LFAST clock

See Figure 29-6 for details.

Address: 0h base + 820h offset = 820h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC1_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 1 Source Selection Control - This value selects the current source for auxiliary clock 1. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 LFAST-SysClk pin

Table continues on the next page...

MC_CGM_AC1_SC field descriptions (continued)

Field	Description
	0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.40 Auxiliary Clock 1 Select Status Register (MC_CGM_AC1_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0h base + 824h offset = 824h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC1_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 1 Source Selection Status - This value indicates the current source for auxiliary clock 1. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 LFAST-SysClk pin 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved

Table continues on the next page...

MC_CGM_AC1_SS field descriptions (continued)

Field	Description
	1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

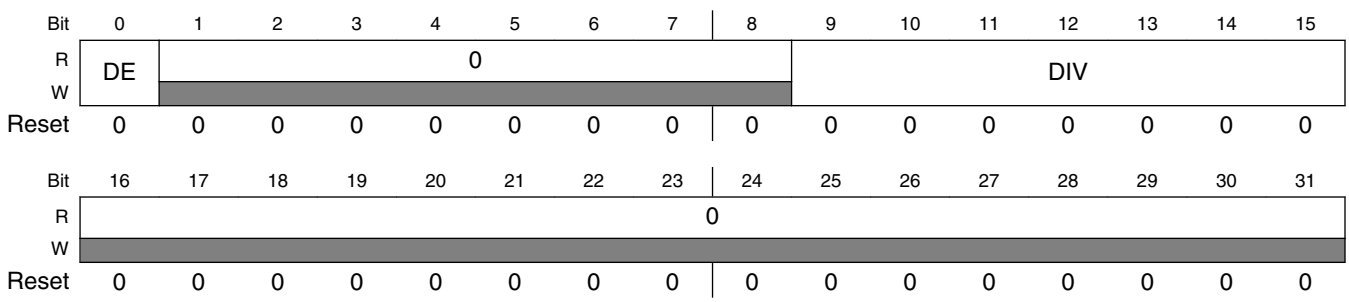
29.3.41 Auxiliary Clock 1 Divider 0 Configuration Register (MC_CGM_AC1_DC0)

This register controls auxiliary clock 0 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 828h offset = 828h



MC_CGM_AC1_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider Division Value - The resultant LFAST clock will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LFAST clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.42 Auxiliary Clock 2 Divider 0 Configuration Register (MC_CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0 when the "CAN jitter" feature is not enabled (see Divider jitter injection).

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 848h offset = 848h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC2_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant FlexRay clock will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FlexRay clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.43 Auxiliary Clock 2 Divider 1 Configuration Register (MC_CGM_AC2_DC1)

This register controls auxiliary clock 2 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 84Ch offset = 84Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC2_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 1 1 Enable auxiliary clock 2 divider 1
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant SENT clock will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the SENT clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.44 Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

See Figure 29-8 for details.

Address: 0h base + 860h offset = 860h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL												0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC3_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_AC3_SC field descriptions (continued)

Field	Description
4–7 SELCTL	Auxiliary Clock 3 Source Selection Control - This value selects the current source for auxiliary clock 3. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.45 Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)

his register provides the current auxiliary clock 3 source selection.

Address: 0h base + 864h offset = 864h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC3_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 3 Source Selection Status - This value indicates the current source for auxiliary clock 3. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 reserved 0011 reserved

Table continues on the next page...

MC_CGM_AC3_SS field descriptions (continued)

Field	Description
	0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.46 Auxiliary Clock 4 Select Control Register (MC_CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

See Figure 29-9 for details.

Address: 0h base + 880h offset = 880h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC4_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 4 Source Selection Control - This value selects the current source for auxiliary clock 4. 0000 reserved 0001 external crystal osc. 0010 reserved 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved

Table continues on the next page...

MC_CGM_AC4_SC field descriptions (continued)

Field	Description
	1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.47 Auxiliary Clock 4 Select Status Register (MC_CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Address: 0h base + 884h offset = 884h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC4_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 4 Source Selection Status - This value indicates the current source for auxiliary clock 4. 0000 reserved 0001 external crystal osc. 0010 reserved 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

Table continues on the next page...

MC_CGM_AC4_SS field descriptions (continued)

Field	Description
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.48 Auxiliary Clock 5 Divider 0 Configuration Register (MC_CGM_AC5_DC0)

This register controls auxiliary clock 5 divider 0. See Fractional divider details for divider characteristics.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8A8h offset = 8A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0		DIV												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 0 1 Enable auxiliary clock 5 divider 0
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–15 DIV	Divider Division Value - The resultant PSI5_f189 clock will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the PSI5_f189 clock remains disabled.
16–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 DIV_FMT	Divider Value Format - This failure indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10

Table continues on the next page...

MC_CGM_AC5_DC0 field descriptions (continued)

Field	Description
	10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

29.3.49 Auxiliary Clock 5 Divider 1 Configuration Register (MC_CGM_AC5_DC1)

This register controls auxiliary clock 5 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8ACh offset = 8ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 1 1 Enable auxiliary clock 5 divider 1
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 DIV	Divider Division Value - The resultant PSI5_f125 clock will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the PSI5_f125 clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

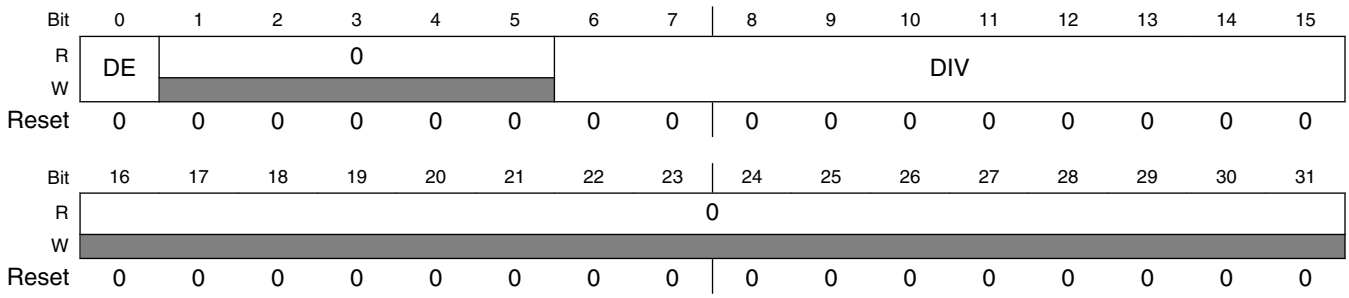
29.3.50 Auxiliary Clock 5 Divider 2 Configuration Register (MC_CGM_AC5_DC2)

This register controls auxiliary clock 5 divider 2.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8B0h offset = 8B0h



MC_CGM_AC5_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 2 1 Enable auxiliary clock 5 divider 2
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 DIV	Divider Division Value - The resultant PSI5_1us clock will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the PSI5_1us clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

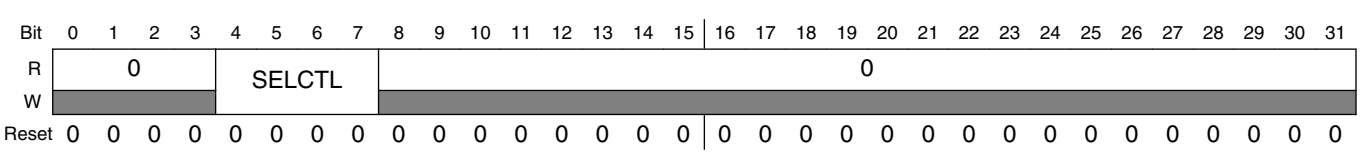
29.3.51 Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 6 divider 0: SYSCLK0 pin clock

See Figure 29-11 for details.

Address: 0h base + 8C0h offset = 8C0h



MC_CGM_AC6_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 6 Source Selection Control - This value selects the current source for auxiliary clock 6. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.52 Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection.

Address: 0h base + 8C4h offset = 8C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC6_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 6 Source Selection Status - This value indicates the current source for auxiliary clock 6. 0000 16 MHz int. RC osc. 0001 external crystal osc.

Table continues on the next page...

MC_CGM_AC6_SS field descriptions (continued)

Field	Description
	0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

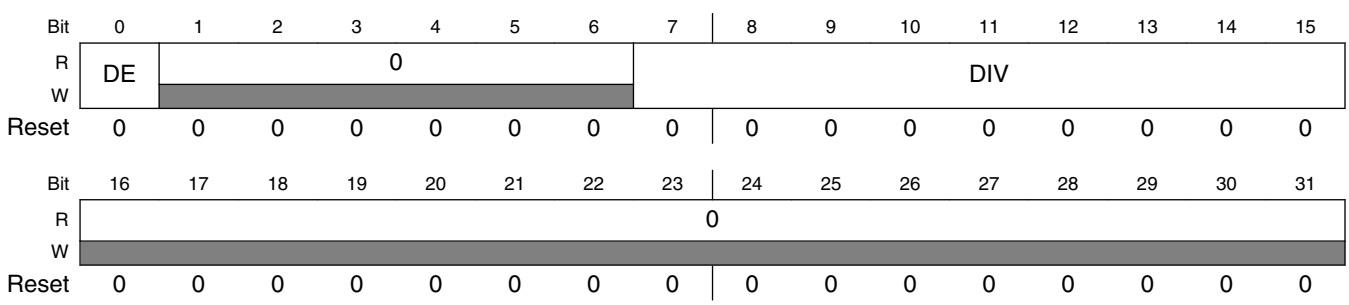
29.3.53 Auxiliary Clock 6 Divider 0 Configuration Register (MC_CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8C8h offset = 8C8h



MC_CGM_AC6_DC0 field descriptions

Field	Description
0 DE	Divider Enable

Table continues on the next page...

MC_CGM_AC6_DC0 field descriptions (continued)

Field	Description
	0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–15 DIV	Divider Division Value - The resultant SYSCLK0 pin clock will have a period 'DIV + 1' times that of auxiliary clock 6. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK0 pin clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.54 Auxiliary Clock 7 Select Control Register (MC_CGM_AC7_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 7 divider 0: SYSCLK1 pin clock

See Figure 29-12 for details.

Address: 0h base + 8E0h offset = 8E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC7_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 7 Source Selection Control - This value selects the current source for auxiliary clock 7. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved

Table continues on the next page...

MC_CGM_AC7_SC field descriptions (continued)

Field	Description
	1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.55 Auxiliary Clock 7 Select Status Register (MC_CGM_AC7_SS)

This register provides the current auxiliary clock 7 source selection.

Address: 0h base + 8E4h offset = 8E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC7_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 7 Source Selection Status - This value indicates the current source for auxiliary clock 7. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.56 Auxiliary Clock 7 Divider 0 Configuration Register (MC_CGM_AC7_DC0)

This register controls auxiliary clock 7 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8E8h offset = 8E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC7_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 7 divider 0 1 Enable auxiliary clock 7 divider 0
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–15 DIV	Divider Division Value - The resultant SYSCLK1 pin clock will have a period 'DIV + 1' times that of auxiliary clock 7. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK1 pin clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.57 Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 8 divider 0: CAN clock

See Figure 29-13 for details.

Address: 0h base + 900h offset = 900h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC8_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 8 Source Selection Control - This value selects the current source for auxiliary clock 8. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.58 Auxiliary Clock 8 Select Status Register (MC_CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Address: 0h base + 904h offset = 904h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC8_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 8 Source Selection Status - This value indicates the current source for auxiliary clock 8. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.59 Auxiliary Clock 8 Divider 0 Configuration Register (MC_CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0 when the "CAN jitter" feature is not enabled (see Divider jitter injection).

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 908h offset = 908h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC8_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 8 divider 0 1 Enable auxiliary clock 8 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant CAN clock will have a period 'DIV + 1' times that of auxiliary clock 8. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.60 Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 9 divider 0: RTI clock

See Figure 29-14 for details.

Address: 0h base + 920h offset = 920h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC9_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 9 Source Selection Control - This value selects the current source for auxiliary clock 9. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved

Table continues on the next page...

MC_CGM_AC9_SC field descriptions (continued)

Field	Description
	1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.61 Auxiliary Clock 9 Select Status Register (MC_CGM_AC9_SS)

This register provides the current auxiliary clock 9 source selection.

Address: 0h base + 924h offset = 924h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC9_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 9 Source Selection Status - This value indicates the current source for auxiliary clock 9. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.62 Auxiliary Clock 9 Divider 0 Configuration Register (MC_CGM_AC9_DC0)

This register controls auxiliary clock 9 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 928h offset = 928h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC9_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 9 divider 0 1 Enable auxiliary clock 9 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant RTI clock will have a period 'DIV + 1' times that of auxiliary clock 9. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the RTI clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.63 Auxiliary Clock 10 Select Control Register (MC_CGM_AC10_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 10 divider 0: ethernet REF clock

See Figure 29-15 for details.

Address: 0h base + 940h offset = 940h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 10 Source Selection Control - This value selects the current source for auxiliary clock 10. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.64 Auxiliary Clock 10 Select Status Register (MC_CGM_AC10_SS)

This register provides the current auxiliary clock 10 source selection.

Address: 0h base + 944h offset = 944h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 10 Source Selection Status - This value indicates the current source for auxiliary clock 10. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.65 Auxiliary Clock 10 Divider 0 Configuration Register (MC_CGM_AC10_DC0)

This register controls auxiliary clock 10 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 948h offset = 948h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 10 divider 0 1 Enable auxiliary clock 10 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value - The resultant ethernet REF clock will have a period 'DIV + 1' times that of auxiliary clock 10. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the ethernet REF clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.4 Functional description

29.4.1 System clock generation

Figure 29-2 shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the 16 MHz int. RC osc. as the system clock and to ignore the system clock select.

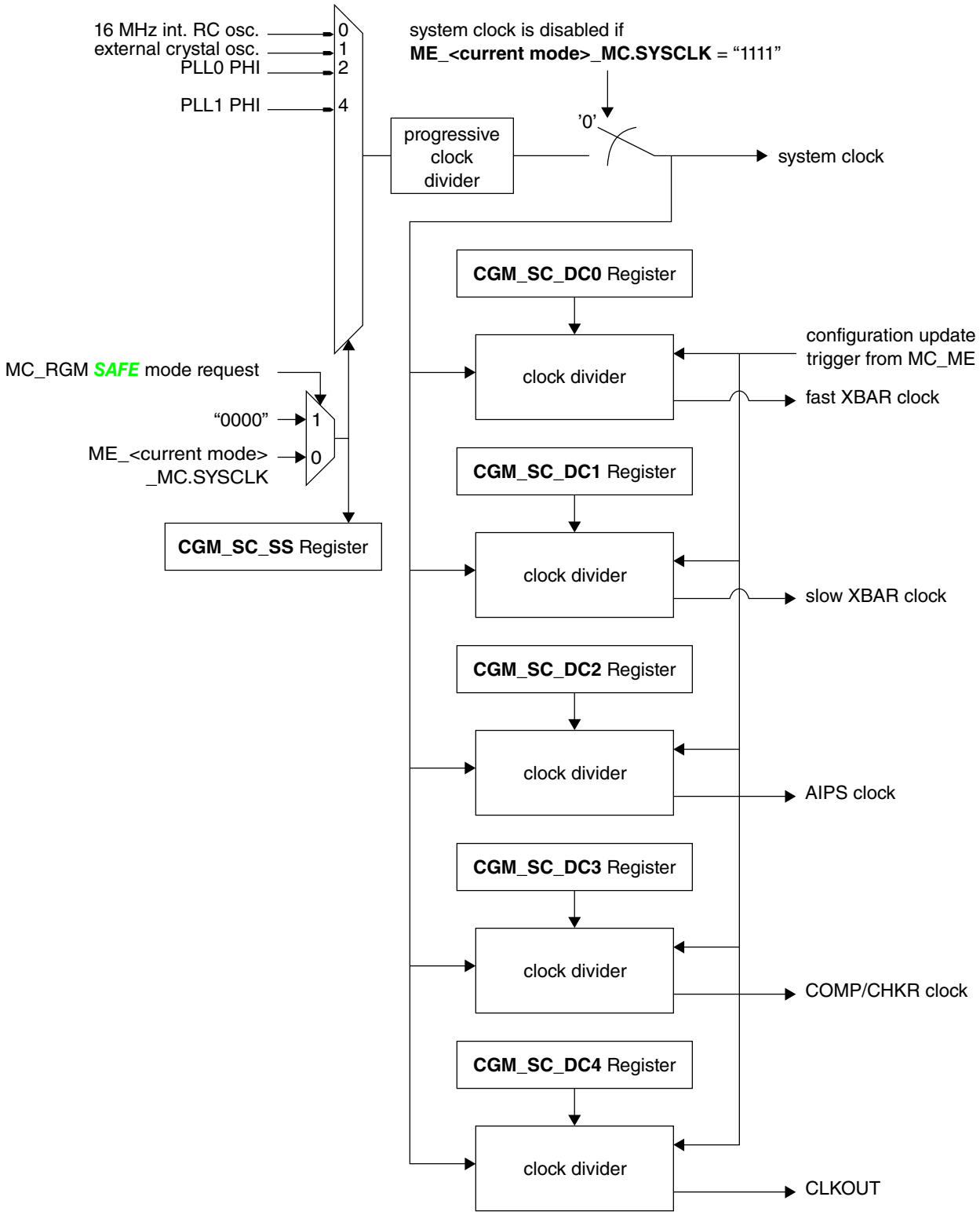


Figure 29-2. MC_CGM system clock generation overview

29.4.1.1 System clock source selection

During normal operation, the system clock selection is controlled

- on a ***SAFE*** mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

29.4.1.2 Progressive system clock switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up based on the power level values of the current and target modes. During ramp-down, the rate of the frequency change is based on the **CGM_PCS_SDUR**, **CGM_PCS_DIVC*n***, and **CGM_PCS_DIVE*n*** registers, where *n* corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the **CGM_PCS_SDUR**, **CGM_PCS_DIVC*n***, and **CGM_PCS_DIVS*n*** registers, where *n* corresponds to the target system clock source selection.

29.4.1.2.1 Generic clock change requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in the following equation.

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

Equation 10. Equation for a_{max}

29.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field **CGM_PCS_SDUR[SDUR]** defines the duration of one system clock switch step in terms of 16 MHz int. RC osc.. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

29.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the **PCS_DIVC*n*[RATE]** register is given in Table 29-1.

Table 29-1. MC_CGM CGM_PCS_DIVC*n*[RATE] values

a_{max}	d = CGM_PCS_DIVC*n*[RATE]
0.05	12
0.10	48
0.15	112
0.20	184

29.4.1.2.4 Generic clock change properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in the following equation.

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

Equation 11. Equation for k

29.4.1.2.5 Clock ramp-down

The clock ramp-down starts with the divider value 1 and with the given divider increment value **PCD_DIVC*n*[RATE]** and ends with the given divider value **PCS_DIVE*n*[DIVE]**.

The divider end value for clock ramp-down is given in the following equation.

$$\text{PCS\_DIVEn}[\text{DIVE}] = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Equation 12

Where f_{curr} is the frequency of the currently selected system clock source.

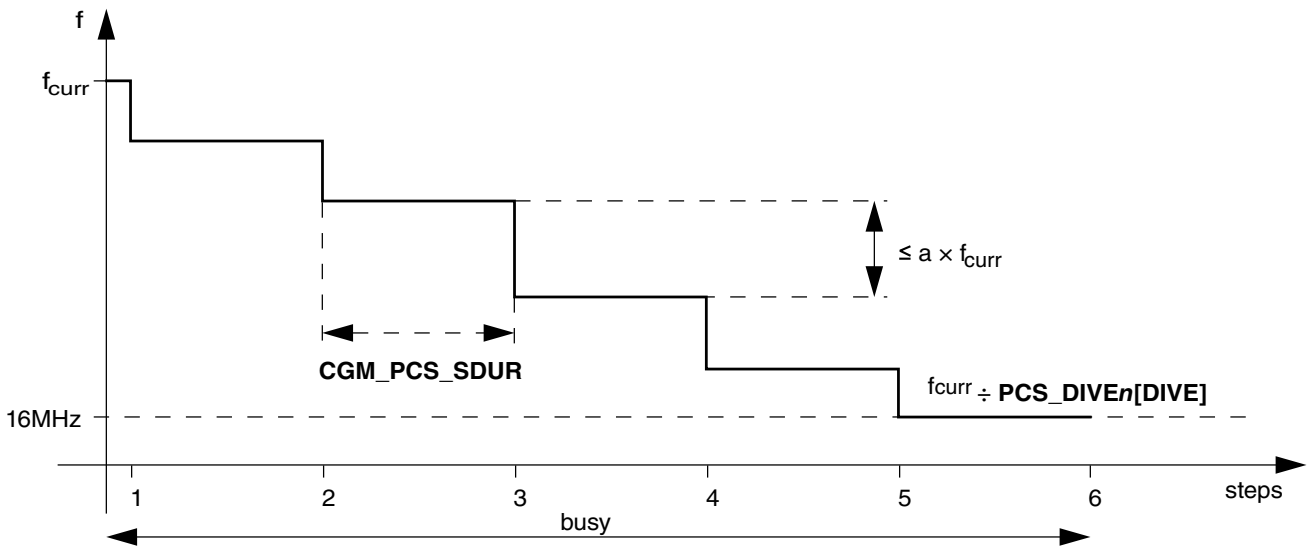


Figure 29-3. MC_CGM System Clock Ramp-Down Timing (k = 6 example)

29.4.1.2.6 Clock ramp-up

The clock ramp-up starts with the given divider value **PCS_DIVS*n*[DIVS]** and with the given divider decrement value **PCD_DIVC*n*[INIT]** and ends with the divider value 1.

The initial divider change start value **PCS_DIVC*n*[INIT]** for system clock ramp-up is given in the following equation.

$$\text{PCS\_DIVCn[INIT]} = d \times k \times 1000$$

Equation 13

Where k is calculated from Equation 11 on page 1204 using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in the following equation.

$$\text{PCS\_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Equation 14

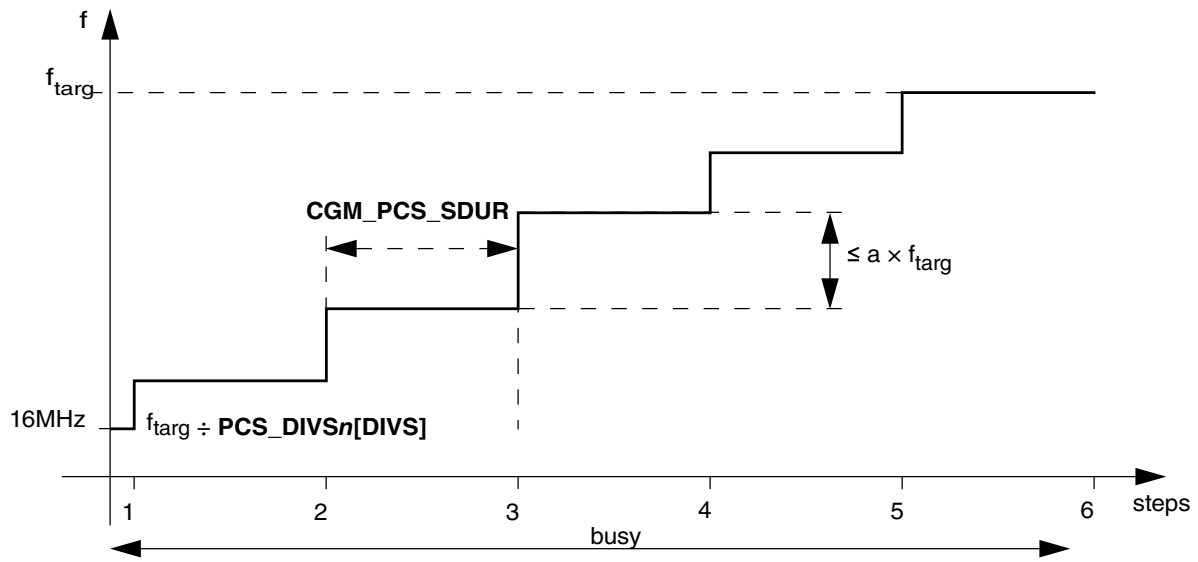


Figure 29-4. MC_CGM System Clock Ramp-Up Timing (k = 6 example)

29.4.1.3 System clock disable

During the TEST mode, the system clock can be disabled by the MC_ME.

29.4.1.4 System clock dividers

The MC_CGM generates the following derived clocks from the system clock:

- fast XBAR clock - controlled by the CGM_SC_DC0 register
- slow XBAR clock - controlled by the CGM_SC_DC1 register
- AIPS clock - controlled by the CGM_SC_DC2 register
- COMP/CHKR clock - controlled by the CGM_SC_DC3 register
- CLKOUT - controlled by the CGM_SC_DC4 register

29.4.1.4.1 System clock divider synchronization

The system clock dividers are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

example

Correct system clock divider configuration

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 2 is an integer multiple of 1, its configuration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

Table 29-2. MC_CGM example system clock division values compatible with divider synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** = 0
1	2	**CGM_SC_DC1[DIV]** = 1
2	6	**CGM_SC_DC2[DIV]** = 5

example

Incorrect system clock divider configuration

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 4 is an integer multiple of 1, its configuration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is not an integer multiple of 4, its configuration is incorrect.

Table 29-3. MC_CGM example system clock division values incompatible with divider synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** = 0
1	4	**CGM_SC_DC1[DIV]** = 3
2	6	**CGM_SC_DC2[DIV]** = 5

warning

Configuring the system clock dividers incorrectly will cause the divided clocks to become un-synchronized. This may lead to lost communication between the clock domains and/or the inability to further change the system clock divider division factors without first disabling them or performing a reset.

NOTE

For 300MHz Core 0 / Core 1 and 200MHz Core 2 / FXBAR operation, the following configuration is allowed:

Table 29-4. Configuration allowed for 300MHz Core 0 / Core 1 and 200MHz Core 2 / FXBAR operation

Divider	Division Factor	Register Value
2	1 2	CGM_ SC_DC 2[DIV] = 11
1	6	CGM_ SC_DC 1[DIV] = 5
0	3	CGM_ SC_DC 0[DIV] = 2
3	2	CGM_ SC_DC 3[DIV] = 1

29.4.1.4.2 Supported External Bus Interfaces clock frequencies

The supported maximum clock frequencies of the External Bus Interface (EBI) and the fast XBAR are as follows:

- EBI clock supported max frequency = 66.67MHz
- fast XBAR clock supported max frequency = 200MHz

The EBI clock is required to be an integer divide relative to the fast XBAR clock, and the CGM dividers need to be programmed correctly to maintain this relationship.

Therefore, the supported combination of fast XBAR and EBI clocks frequencies are as follows:

Table 29-5. Possible combination of fast XBAR and EBI clock frequencies

Ratio	Fast XBAR and EBI clock frequencies
1:1	66.67MHz:66.67MHz, 50MHz:50MHz, and 16MHz:16MHz
2:1	100MHz:50MHz
3:1	200MHz:66.67MHz (normal use case)
4:1	200MHz:50MHz

29.4.1.4.3 System clock divider update triggering

As opposed to the auxiliary clock dividers, which update immediately on a configuration change, the system clock dividers are updated only with the next software triggered mode change. This is to ensure that the system clock dividers are updated simultaneously and that no intermediate inconsistent configurations can occur. It also ensures that the divider configurations are aligned with the target mode's system clock selection.

The system clock divider configuration update sequence is as follows:

1. Set the CGM_DIV_UPD_TYPE, SYS_UPD_TYPE field as "1" to enable the update via a software trigger.
2. Configure the **CGM_SC_DC0…2** registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in System clock divider synchronization.
3. **Optional Step:** If the divider ratios are not changing with respect to the previous configuration, the CGM_SC_DIV_RC, SYS_DIV_RATIO_CHNG field may be reset to 0 to disable the crossbar halt handshake mechanism.
4. Write any non-zero value to the CGM_DIV_UPD_TRIG register to trigger the divider update.

See Mode Entry Module (MC_ME) chapter for details on how to configure modes and make mode change requests.

29.4.2 Auxiliary clock generation

Figure 29-5 through Figure 29-15 show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)
- Auxiliary Clock 1 Select Control Register (MC_CGM_AC1_SC)
- Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)
- Auxiliary Clock 4 Select Control Register (MC_CGM_AC4_SC)
- Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)
- Auxiliary Clock 7 Select Control Register (MC_CGM_AC7_SC)
- Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)
- Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)
- Auxiliary Clock 10 Select Control Register (MC_CGM_AC10_SC)

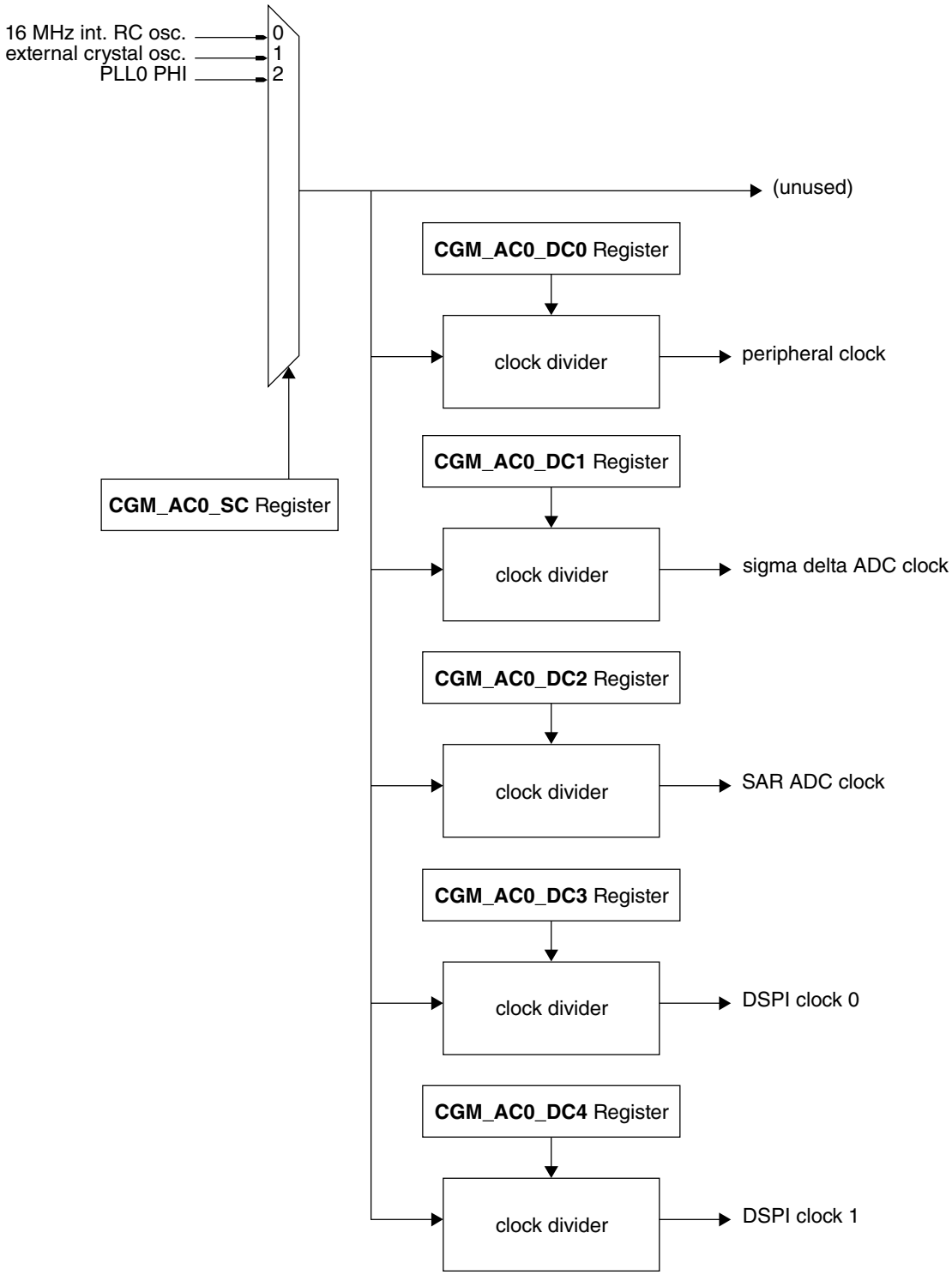


Figure 29-5. MC_CGM Auxiliary Clock 0 Generation Overview

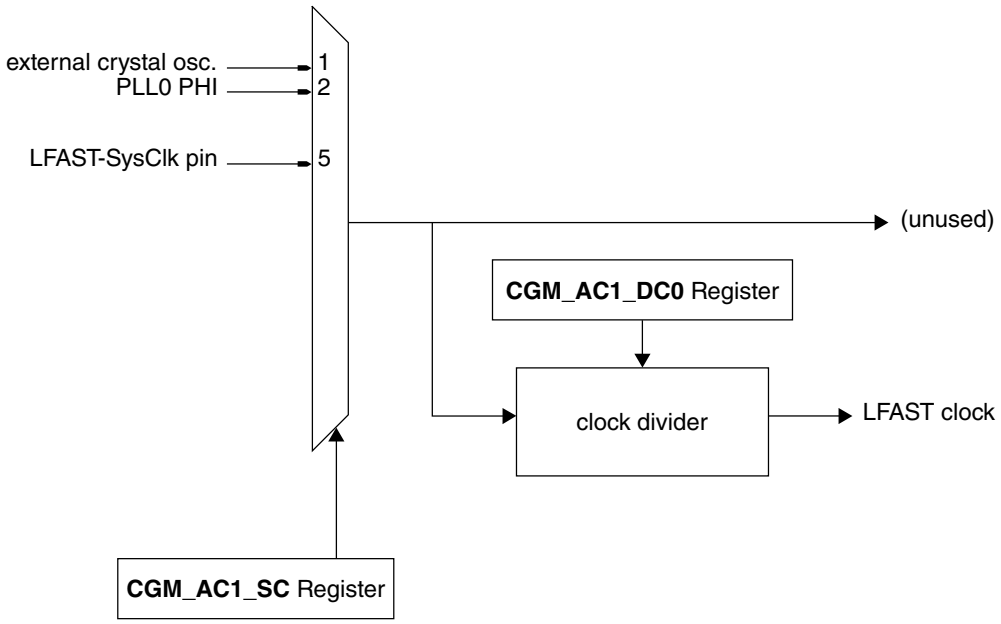


Figure 29-6. MC_CGM Auxiliary Clock 1 Generation Overview

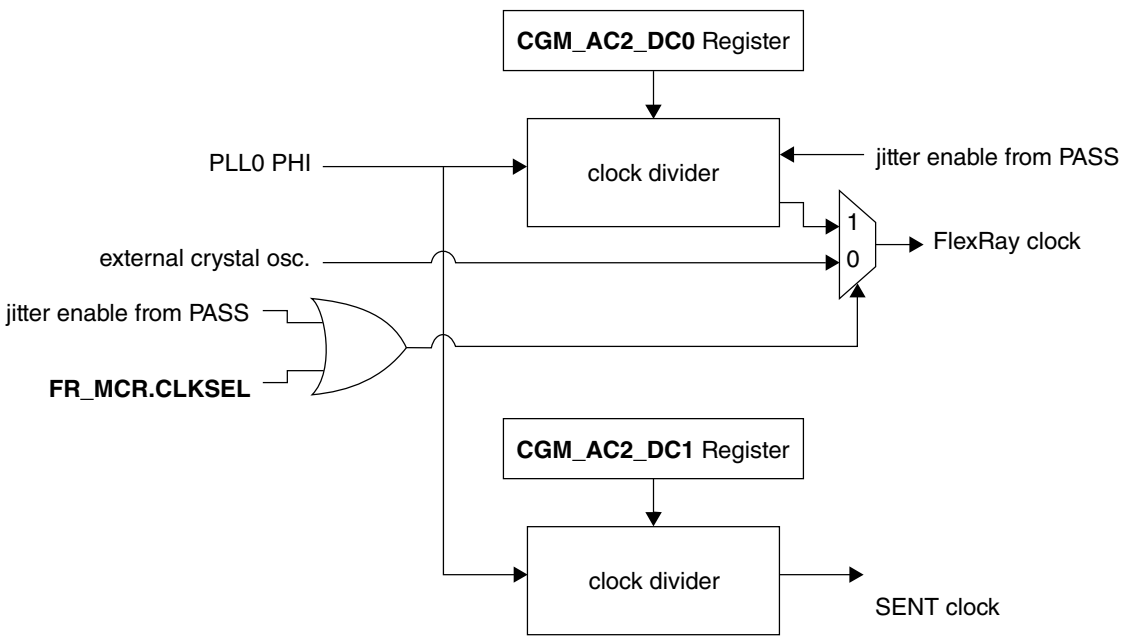


Figure 29-7. MC_CGM Auxiliary Clock 2 Generation Overview

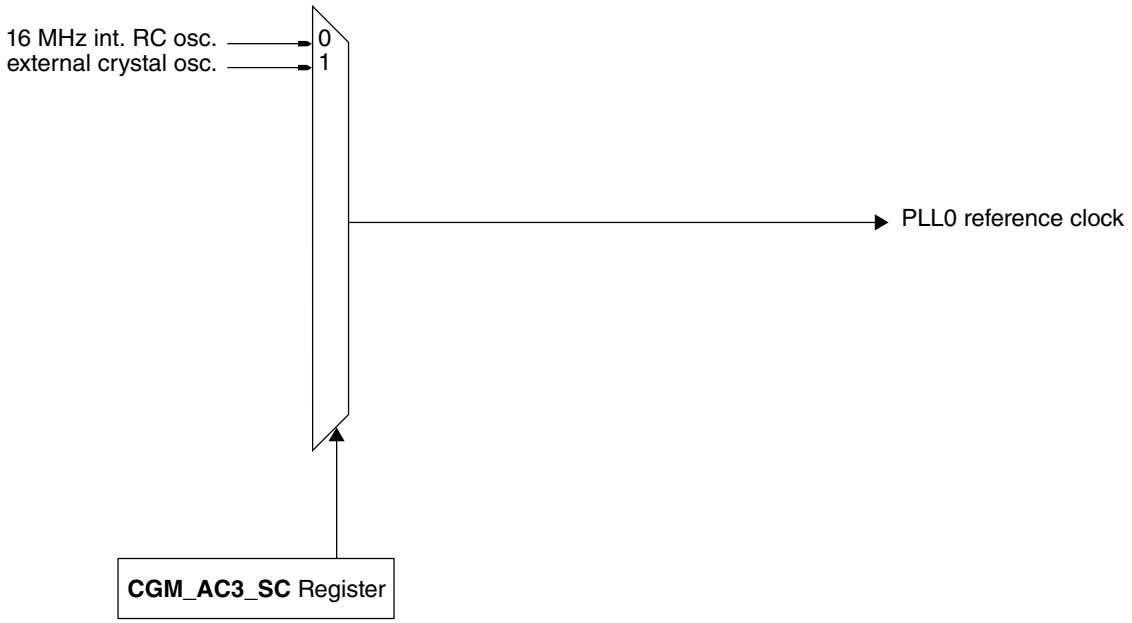


Figure 29-8. MC_CGM Auxiliary Clock 3 Generation Overview

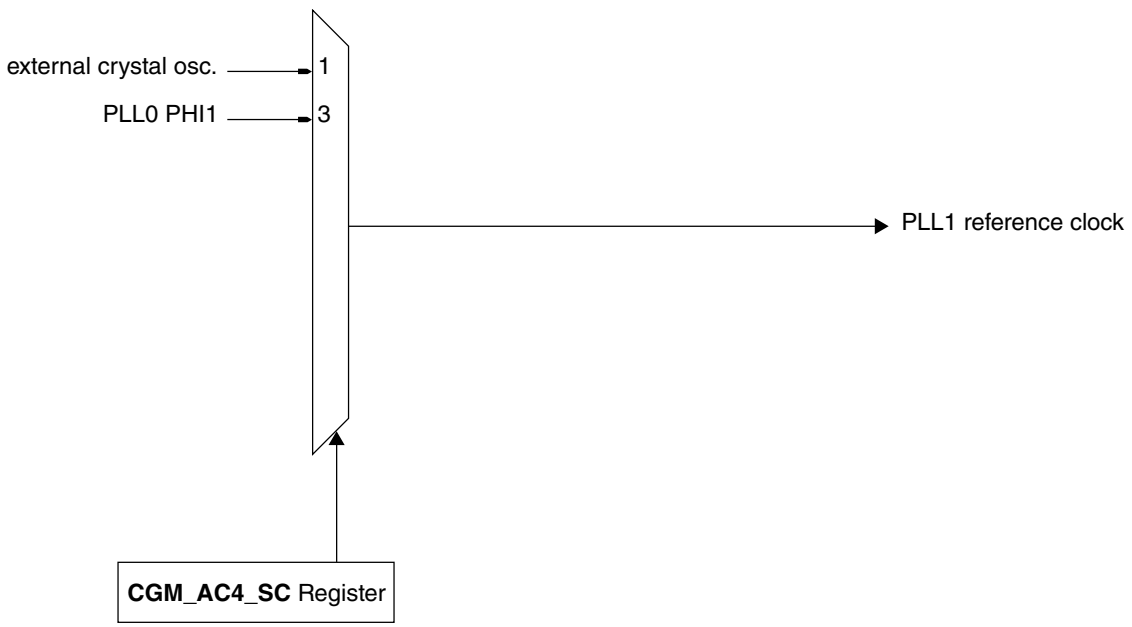


Figure 29-9. MC_CGM Auxiliary Clock 4 Generation Overview

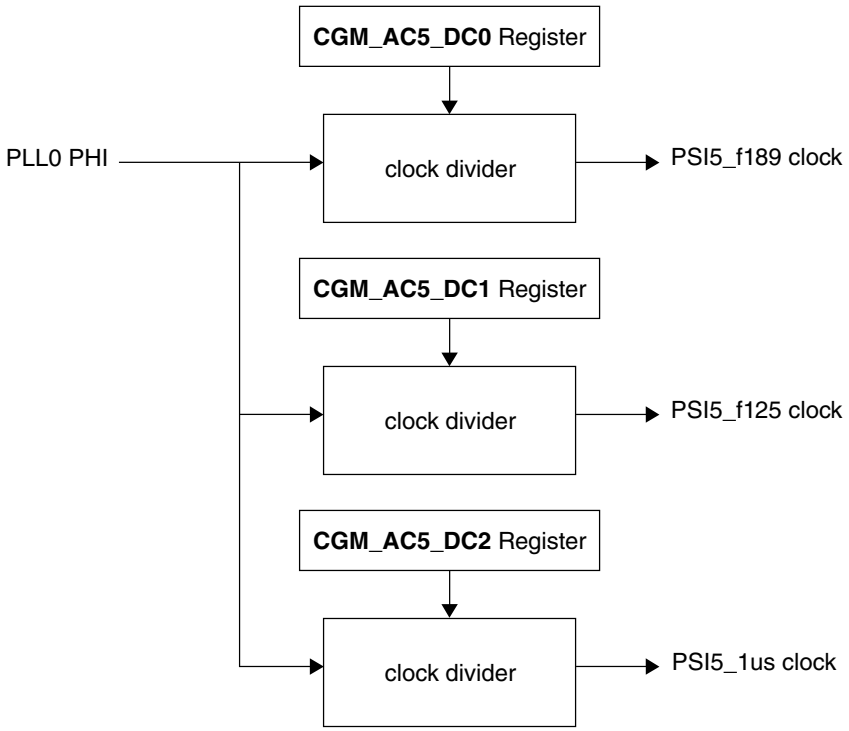


Figure 29-10. MC_CGM Auxiliary Clock 5 Generation Overview

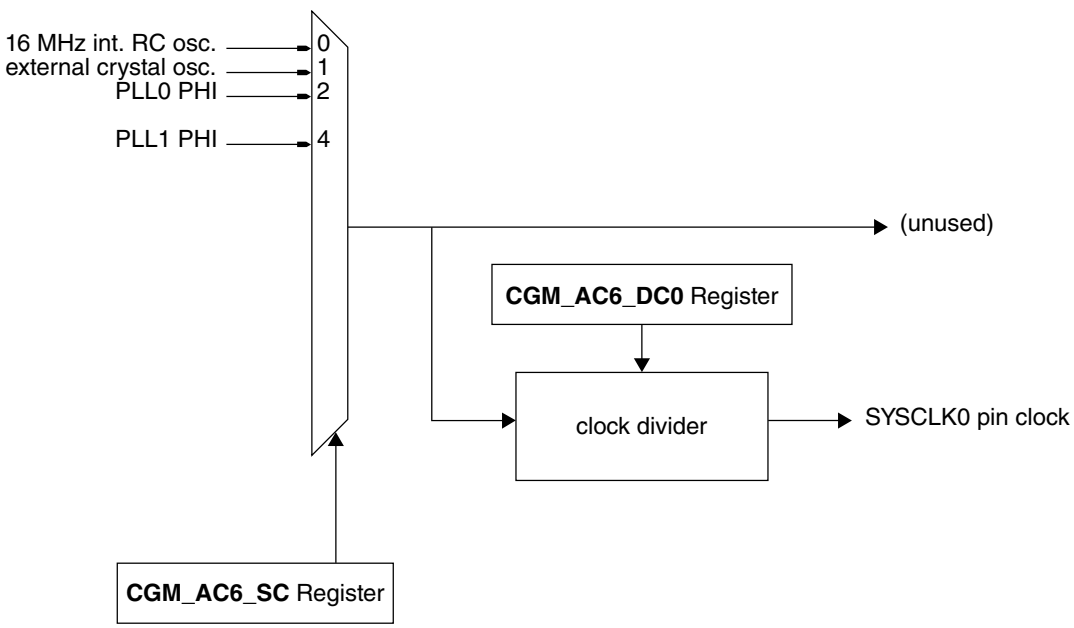


Figure 29-11. MC_CGM Auxiliary Clock 6 Generation Overview

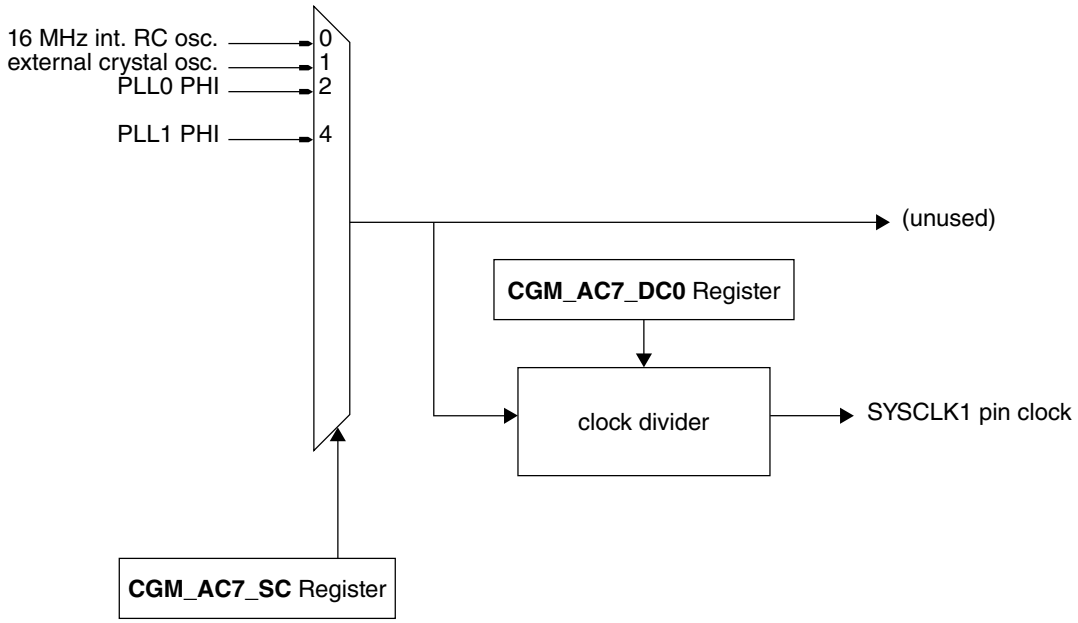


Figure 29-12. MC_CGM Auxiliary Clock 7 Generation Overview

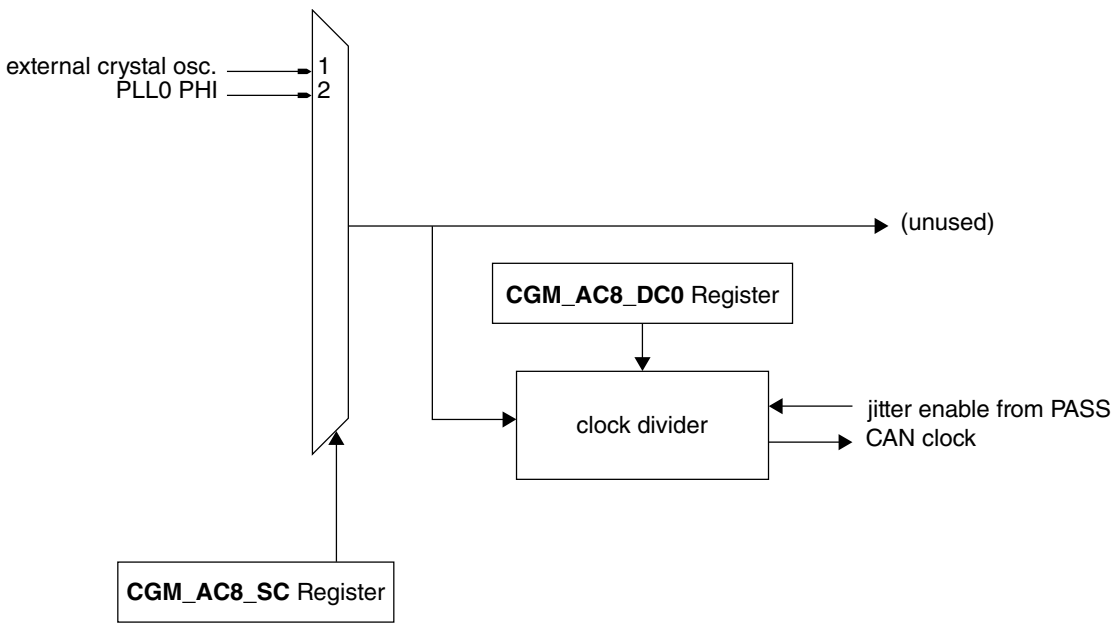


Figure 29-13. MC_CGM Auxiliary Clock 8 Generation Overview

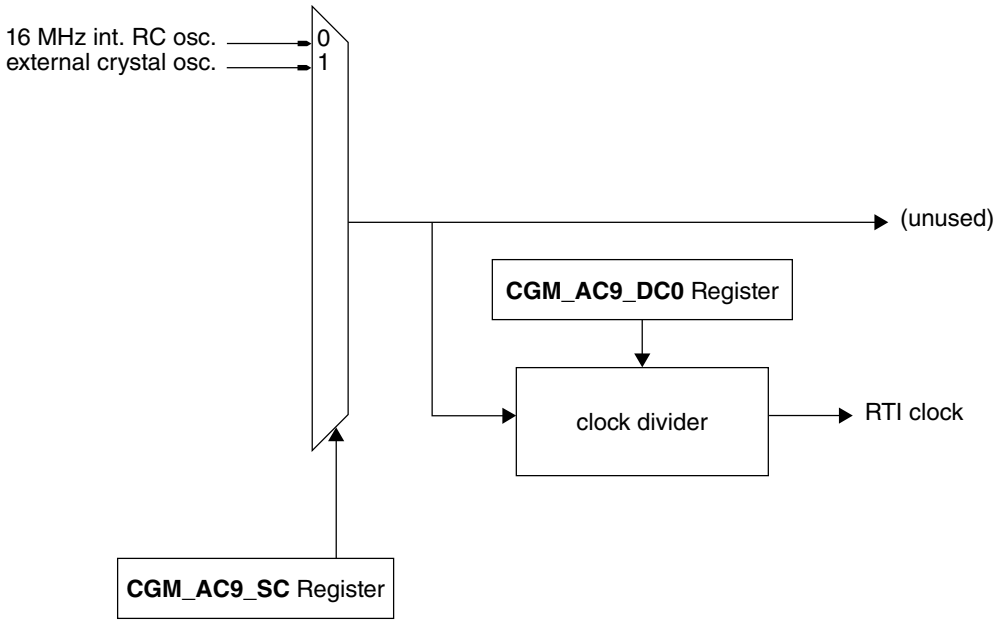


Figure 29-14. MC_CGM Auxiliary Clock 9 Generation Overview

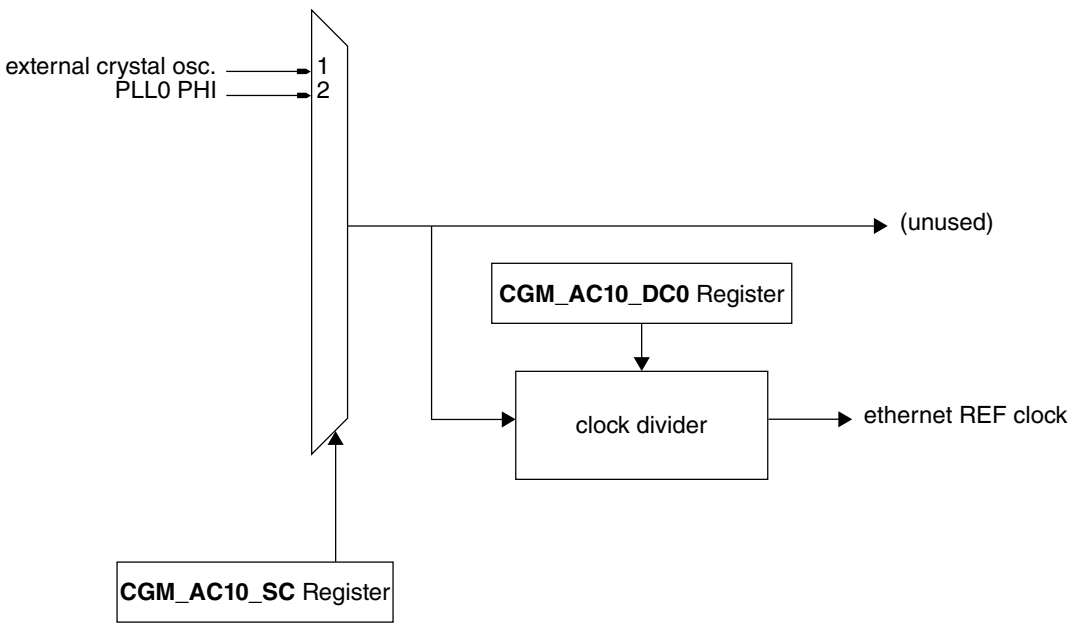


Figure 29-15. MC_CGM Auxiliary Clock 10 Generation Overview

29.4.2.1 Auxiliary clock dividers

The MC_CGM generates the following derived clocks:

- peripheral clock - controlled by the CGM_AC0_DC0 register
- sigma delta ADC clock - controlled by the CGM_AC0_DC1 register

- SAR ADC clock - controlled by the CGM_AC0_DC2 register
- DSPI clock 0 - controlled by the CGM_AC0_DC3 register
- LFAST clock - controlled by the CGM_AC1_DC0 register
- FlexRay clock - controlled by the CGM_AC2_DC0 register
- SENT clock - controlled by the CGM_AC2_DC1 register
- PSI5_f189 clock - controlled by the CGM_AC5_DC0 register
- PSI5_f125 clock - controlled by the CGM_AC5_DC1 register
- PSI5_1us clock - controlled by the CGM_AC5_DC2 register
- SYSCLK0 pin clock - controlled by the CGM_AC6_DC0 register
- SYSCLK1 pin clock - controlled by the CGM_AC7_DC0 register
- CAN clock - controlled by the CGM_AC8_DC0 register
- RTI clock - controlled by the CGM_AC9_DC0 register
- ethernet REF clock - controlled by the CGM_AC10_DC0 register

29.4.3 Dividers functional description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)
- System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)
- System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)
- System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)
- System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)
- Auxiliary Clock 0 Divider 0 Configuration Register (MC_CGM_AC0_DC0)
- Auxiliary Clock 0 Divider 1 Configuration Register (MC_CGM_AC0_DC1)
- Auxiliary Clock 0 Divider 2 Configuration Register (MC_CGM_AC0_DC2)
- Auxiliary Clock 0 Divider 3 Configuration Register (MC_CGM_AC0_DC3)
- Auxiliary Clock 0 Divider 4 Configuration Register (MC_CGM_AC0_DC4)

- Auxiliary Clock 1 Divider 0 Configuration Register (MC_CGM_AC1_DC0)
- Auxiliary Clock 2 Divider 0 Configuration Register (MC_CGM_AC2_DC0)
- Auxiliary Clock 2 Divider 1 Configuration Register (MC_CGM_AC2_DC1)
- Auxiliary Clock 5 Divider 0 Configuration Register (MC_CGM_AC5_DC0)
- Auxiliary Clock 5 Divider 1 Configuration Register (MC_CGM_AC5_DC1)
- Auxiliary Clock 5 Divider 2 Configuration Register (MC_CGM_AC5_DC2)
- Auxiliary Clock 6 Divider 0 Configuration Register (MC_CGM_AC6_DC0)
- Auxiliary Clock 7 Divider 0 Configuration Register (MC_CGM_AC7_DC0)
- Auxiliary Clock 8 Divider 0 Configuration Register (MC_CGM_AC8_DC0)
- Auxiliary Clock 9 Divider 0 Configuration Register (MC_CGM_AC9_DC0)
- Auxiliary Clock 10 Divider 0 Configuration Register (MC_CGM_AC10_DC0)

The reset value of all counters is '1'. If a divider has its **DE** bit in the respective configuration register set to '0' (the divider is disabled), any value in its **DIV** field is ignored.

29.4.3.1 Divider jitter injection

The clock dividers for the FlexRay clock and the CAN clock inject a jitter so that off-chip communication via the associated peripherals is not possible. For details on enabling this feature, see the "Production Disable" and "CAN Jitter Enable" sections of the PASS chapter.

In addition, PLL0 PHI is always selected to source the FlexRay clock when jitter injection is enabled (see Figure 29-7).

29.4.3.2 Fractional divider details

The clock dividers used to generate the PSI5_f189 clock and the DSPI clock 0 divide the selected clock source frequency by a fractional value given by the **DIV** and **DIV_FMT** fields of the **CGM_AC5_DC0** register.

$$F_{\text{divided clock}} = F_{\text{source clock}} \div \left((\text{DIV} + 1) \div 10^{\text{DIV\_FMT}} \right)$$

Equation 15. Equation for $F_{\text{divided clock}}$

The divided clock frequency is actually an average frequency, since the divider uses only the available rising and falling edges of the source clock. Therefore, the actual frequency switches between two frequencies which are the two frequencies closest to the desired frequency (above and below) which can be extracted by a division factor $n/2$, where n is an integer greater than 1. The switching between the two frequencies is such that the resultant clock's rising edge never deviates from the ideal clock's rising edge by more than 25% of the source clock period and that the resultant average frequency over **DIV** + 1 source clock periods (= $10^{\mathbf{DIV\_FMT}}$ divided clock periods) is the desired frequency.

example

Fractional clock division

Desired division factor = 2.3

DIV = 22

DIV_FMT = 1

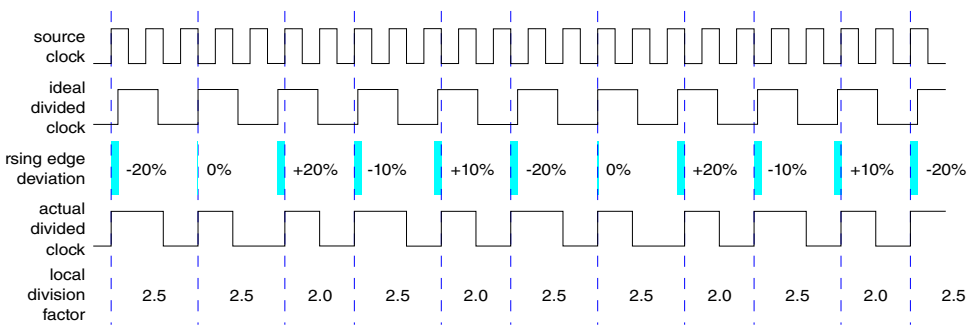


Figure 29-16. MC_CGM fractional division example waveform (divide by 2.3)

$$F_{\text{divided clock}}\big|_{\text{max}} = F_{\text{source clock}} \div 2.0$$

Equation 16. Equation for $F_{\text{divided clock, max}}$

$$F_{\text{divided clock}}\big|_{\text{min}} = F_{\text{source clock}} \div 2.5$$

Equation 17. Equation for $F_{\text{divided clock, min}}$

$$F_{\text{divided clock}}\big|_{\text{ave}} = F_{\text{source clock}} \div \frac{((2.5\times 6)+(2.0\times 4))}{10}$$

Equation 18. Equation for $F_{\text{divided clock, ave}}$

Therefore,

$$F_{\text{divided clock}}\big|_{\text{ave}} = F_{\text{source clock}} \div 2.3$$

Equation 19. Equation for $F_{\text{divided clock, ave}}$

As can be seen in Figure 29-16, the rising edge of the actual divided clock is 'snapped' to the rising or falling edge of the source clock closest to the rising edge of the ideal divided clock. This results in a theoretical maximum deviation, or jitter, of ±25% of the source clock period. In this particular case, the maximum deviation is ±20% of the source clock period.

29.4.3.3 Divider update triggering

There are two methods for updating the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the CGM_DIV_UPD_TYPE register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the CGM_UPD_STAT register is set. This bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

29.4.3.3.1 Immediate divider update

If a divider group has been configured to be updated immediately (i.e., the corresponding CGM_DIV_UPD_TYPE register bit = '0'), the configuration of a divider in that group will be applied immediately on the write to that divider’s configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

29.4.3.3.2 Software-triggered divider update

If a divider group has been configured to be updated via a software trigger (i.e., the corresponding CGM_DIV_UPD_TYPE register bit = '1'), the configuration of a divider in that group will not be applied immediately on the write to that divider’s configuration register. Instead, the configuration in the corresponding configuration register will be applied only when software writes any value to the CGM_DIV_UPD_TRIG register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

In the case of the system clock dividers, if the SYS_DIV_RATIO_CHNG bit in the CGM_SC_DIV_RC register = '1', the crossbar switch is requested to halt its ongoing transactions, and only after the crossbar switch has acknowledged the request does the update of the system clock dividers take place. Once these updates have completed, the halt request to the crossbar is deasserted.

NOTE

If halting the crossbar can cause a bus master to starve and thus lead to lost data or other problems, it is recommended to vary the system clock divider configurations without changing the ratios and to set CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] to '0'.

NOTE

In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (i.e., all relevant bits in the CGM_DIV_UPD_STAT register are ‘0’) before triggering a new update via the CGM_DIV_UPD_TRIG register.

The software-triggered divider update sequence for system clocks is as follows:

1. Set the CGM_DIV_UPD_TYPE, SYS_UPD_TYPE field as "1" to enable the update via a software trigger.
2. Configure the **CGM_SC_DC0…2** registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in System clock divider synchronization.
3. **Optional Step:** If the divider ratios are not changing with respect to the previous configuration, the CGM_SC_DIV_RC, SYS_DIV_RATIO_CHNG field may be reset to 0 to disable the crossbar halt handshake mechanism.
4. Write any non-zero value to the CGM_DIV_UPD_TRIG register to trigger the divider update.

Poll the status register CGM_DIV_UPD_STAT, SYS_UPD_STAT field to check when the divider update completes.

Chapter 30
OSC Digital Interface (XOSC)

30.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provide the register interface for the programmable features. Selection of the XOSC as a source clock is controlled by the Clock Generation Module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry Module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control

30.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that is used as an input to the PLLs and can be selected as the reference clock for many of the peripherals on the device.

The table below shows the possible configurations of the XOSC.

Table 30-1. XOSC configurations

OSC EN (from MC_ME)	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC Output	XOSC MODE
0	0	Crystal, or left opened	Crystal, or left opened	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

30.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see "Mode Entry Module (MC_ME)" chapter). The ME_<*mode*>_MC[XOSCON] controls the power down of oscillator based on the current device mode (see "MC_ME Memory Map" table for available *modes*). The status of the XOSC is shown in the MC_ME_GS[S_XOSC].

30.2.2 Oscillator startup delay

A bit in the UTEST flash memory determines whether the analog portion of the oscillator is enabled at powerup. If enabled, the analog portion of the oscillator is powered-up during Phase3 of the device reset sequence (pre-self-test sequence). In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to PLL0).

The default value for this field after reset depends on implementation and can be found in the XOSC Device Configuration. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note

It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

30.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = XOSC_CTL[EOCV] × 512). The XOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (XOSC_CTL[M_OSC] = 1).

30.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This field can only be set by software and is only cleared by a system reset. In bypass mode, the oscillator is disabled and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

NOTE

The external oscillator must be running before the XOSC is turned off. If XOSC is being turned on, the status will be updated only after the clock starts toggling.

NOTE

Power down from the MC_ME will have no effect on the oscillator. The bypass clock will continue running in the device. It is ensured that mode transitions from the MC_ME complete successfully by providing bypass information to the MC_ME.

30.3 Memory map and register definition

XOSC memory map

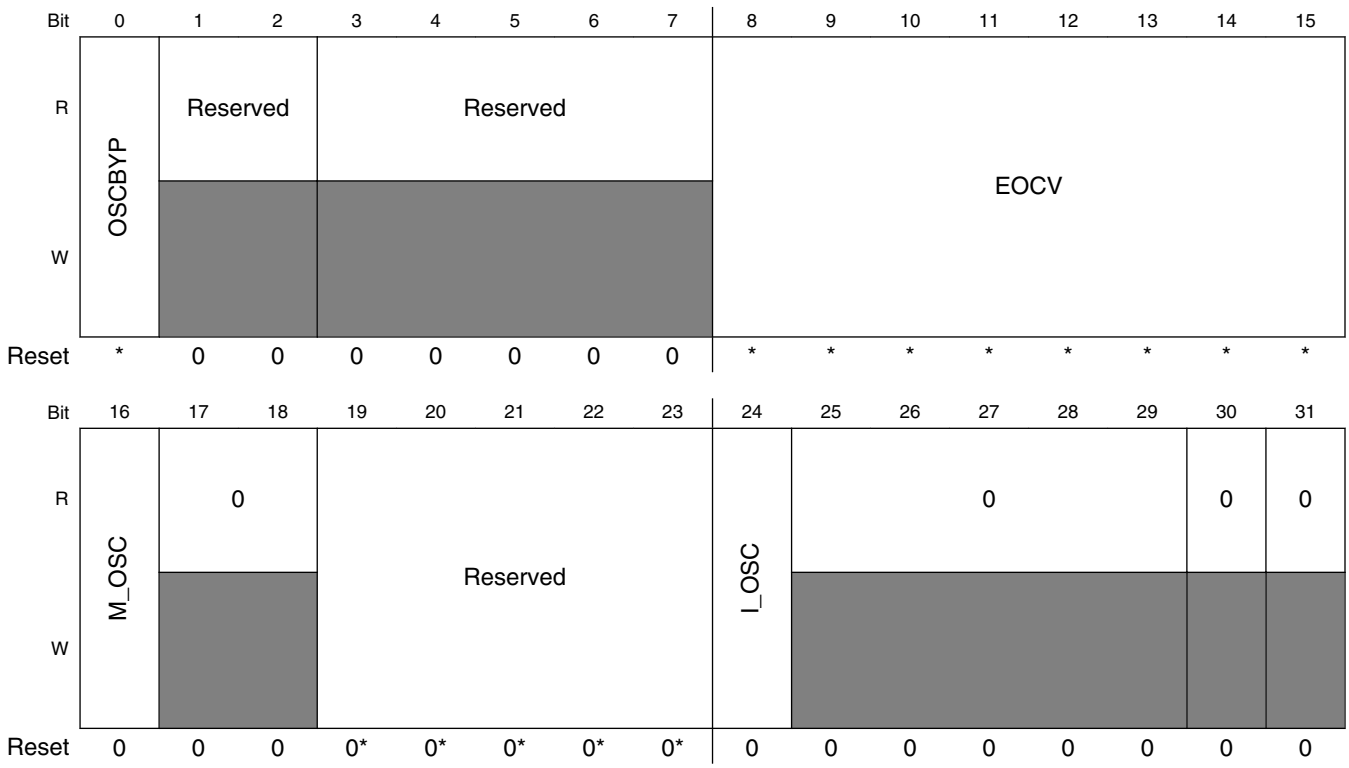
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XOSC Control Register (XOSC_CTL)	32	R/W	See section	30.3.1/1226

30.3.1 XOSC Control Register (XOSC_CTL)

NOTE

The XOSC_CTL is only writable in supervisor mode.

Address: 0h base + 0h offset = 0h



* Notes:

- Reserved field: This field can be written at any time, but writes have no effect. Reads return the previously written value.
- M_OSC field: See Clocking chapter for reset value.
- EOCV field: See Clocking chapter for reset value.
- OSCBYP field: See Clocking chapter for reset value.

XOSC_CTL field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
1–2 Reserved	This field is reserved.

Table continues on the next page...

XOSC_CTL field descriptions (continued)

Field	Description
3–7 Reserved	This field is reserved.
8–15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV × 512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24 I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when OSCCNT counter reaches the count value EOCV x 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.
25–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 31
IRCOSC Digital Interface

31.1 Introduction

The Internal RC Oscillator Digital Interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC), as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage. It also provides access to trimming information used for safety implementations.

31.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see the Data Sheet for timing details) which is required for stabilization. The IRCOSC status is updated in the MC_ME_GS[S_IRC] field (see the "Mode Entry Module MC_ME" chapter).

The IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

31.3 Memory map and register definition

IRCOSC memory map

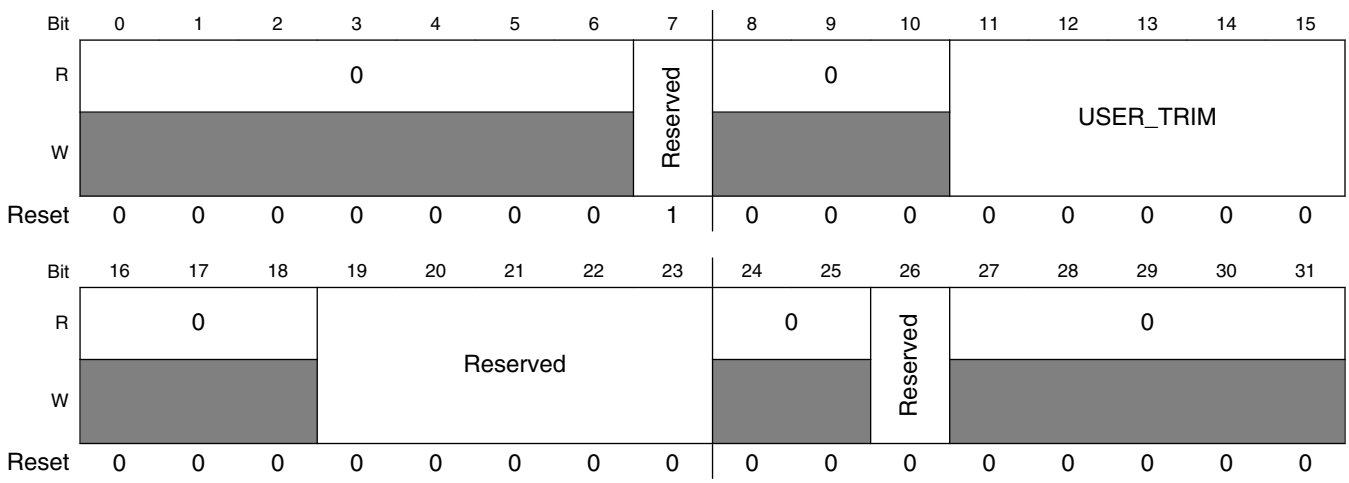
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	See section	31.3.1/1230

31.3.1 IRCOSC Control Register (IRCOSC_CTL)

The IRCOSC_CTL contains user programmable parameters.

IRCOSC_CTL is writable only in supervisor mode.

Address: 0h base + 0h offset = 0h



IRCOSC_CTL field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This bit can be written with any value, but writes are ignored. A read returns last written value.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 USER_TRIM	User trimming bits with respect to nominal factory frequency The MSB of the USER_TRIM bits is used to determine whether the frequency will be increased or decreased. If MSB = 0 then a change in USER_TRIM[3:0] will decrease the frequency, and likewise, if MSB = 1 then a change in USER_TRIM[3:0] will increase the frequency. Frequency trimming calculation shows how RCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

IRCOSC_CTL field descriptions (continued)

Field	Description
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This field may sometimes have the value of 1, and can be cleared by writing a 1 to this bit. However, this field has no effect on the IRCOSC or chip behavior.
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.2 Frequency trimming calculation

Table 31-1. IRCOSC_CTL[USER_TRIM] frequency trimming calculation

USER_TRIM[4:0] value	Frequency
....	....
10011	$F + (3 \times \delta F_{var\_SW})$
10010	$F + (2 \times \delta F_{var\_SW})$
10001	$F + \delta F_{var\_SW}$
10000	F
00000	F
00001	$F - \delta F_{var\_SW}$
00010	$F - (2 \times \delta F_{var\_SW})$
00011	$F - (3 \times \delta F_{var\_SW})$
....	....

NOTE

See the data sheet for δF_{var_SW} value and min/max frequency variations.

Chapter 32
RAM Controller (PRAMC)

32.1 Introduction

This section provides an overview of the Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports two 64-bit AHB interfaces and a 64-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access. Shown below in Figure 32-1 is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture.

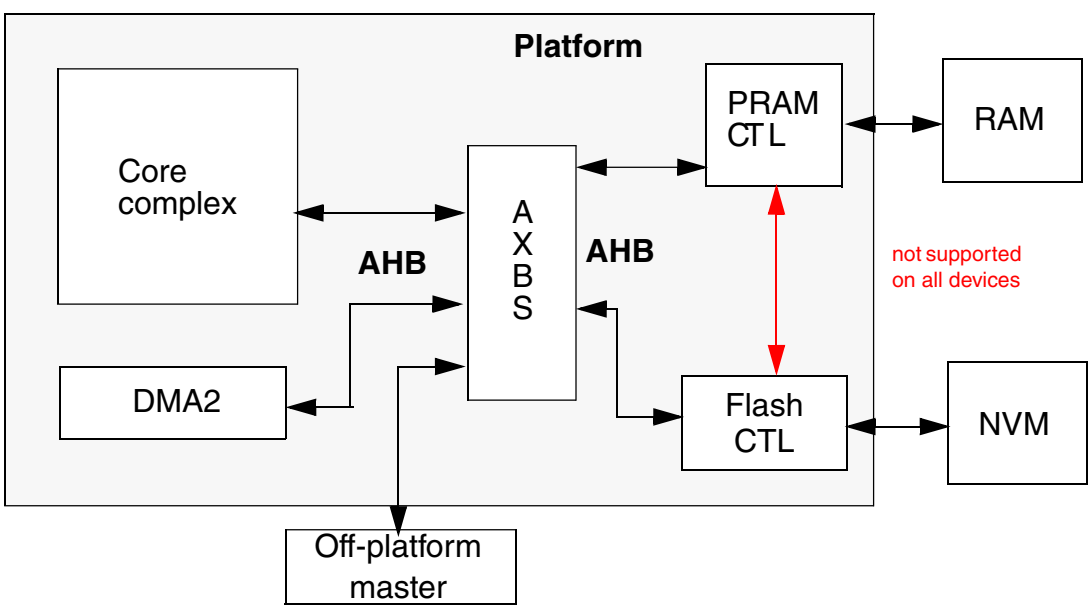


Figure 32-1. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a doubleword

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to present simultaneously the address and associated write data, which is not valid until the AHB dataphase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read datapath contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read databus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write datapath contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

32.2 SRAM controller memory map and register definition

The RAM controller module provides an IPS programming model mapped to a standard on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.

NOTE

Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software must be architected to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, e.g., DMA or communications modules, are enabled. Move any instruction execution or memory references outside the system RAM while updating the PRAMC configuration, e.g., to the core local memory space.

PRAMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32	R/W	0000_0200h	32.2.1/1235

32.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PRI		P1_BO_DIS	P0_BO_DIS	P1_RB_DIS	P0_RB_DIS	0			FT_DIS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

PRAMC_PRCR1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PRAMC_PRCR1 field descriptions (continued)

Field	Description
22–23 PRI	AHB port arbitration mode. Use this register to select the AHB port arbitration mode. 00 Round Robin arbitration is selected. 01 Port p0 has priority over port p1. 10 Port p1 has priority over port p0. 11 Reserved.
24 P1_BO_DIS	Port p1 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
26 P1_RB_DIS	Port p1 read buffer disable. 0 Read buffer enabled and is used for servicing sequential, less-than-4-bit read requests. 1 Read buffer disabled and invalidated and all read requests generate array accesses.
27 P0_RB_DIS	Port p0 read buffer disable. 0 Read buffer enabled and is used for servicing sequential, less than 64-bit read requests. 1 Read buffer disabled and invalidated and all read requests generate array accesses.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset.

Table continues on the next page...

PRAMC_PRCR1 field descriptions (continued)

Field	Description
	NOTE: Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

32.3 Functional description

This section describes the functions of the RAM controller.

32.3.1 Read / Write introduction

The RAM controller generates chip-select and write-enable, the array address, and write data as inputs to the RAM array. The RAM controller captures read data from the RAM array and drives it onto the AHB along with the associated dataphase termination signals. To facilitate operating at higher frequencies, the RAM controller can optionally incur a single wait state on the AHB response.

32.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the AHB.

32.3.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCRx[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCRx[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCRx[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCRx[FT_DIS]=1, and a WRAP4 burst will have an access time of 3-2-2-2.

NOTE

The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency.

32.3.2 Writes

This section discusses various write operations of the RAM controller.

32.3.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back to back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

32.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. Figure 32-2 provides details on the read-modify-write datapath. For details on the ECC coding scheme, refer to Reliability considerations.

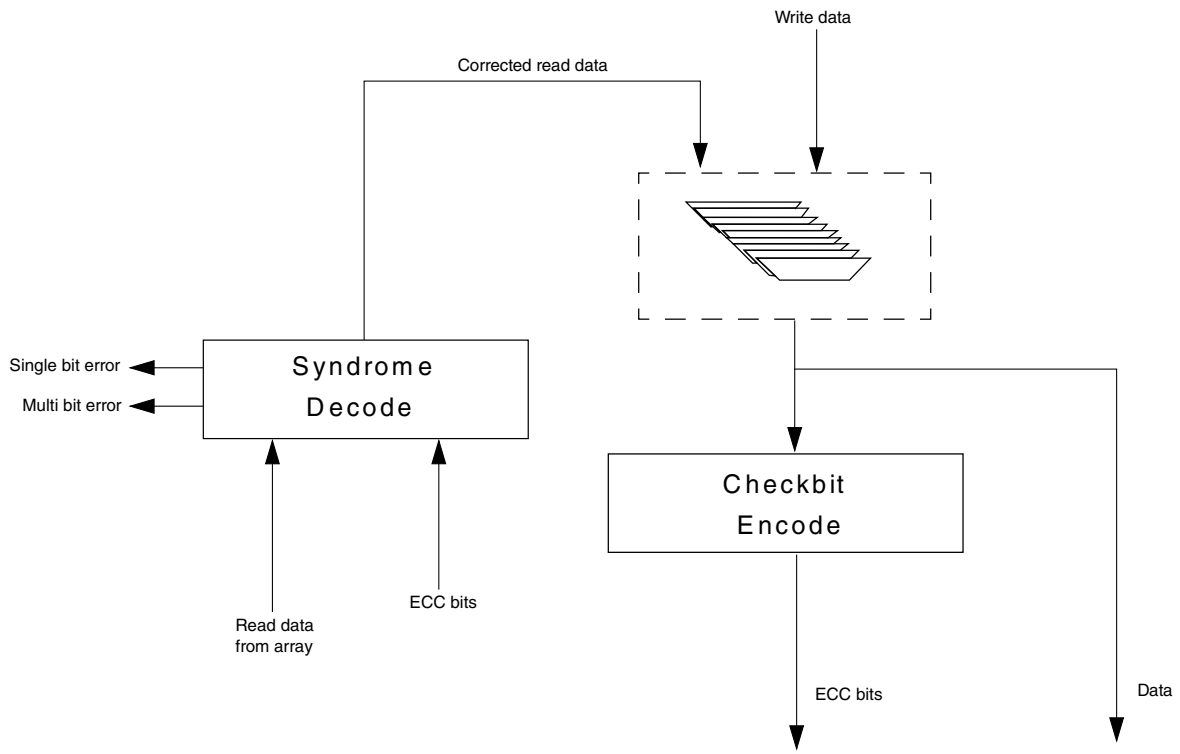


Figure 32-2. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

32.3.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

NOTE

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

32.4 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

32.5 Reliability considerations

This section discusses reliability considerations of the RAM controller.

32.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR'ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 32-1. RAM controller basic H-matrix definition

		XOR sum	Data Bit[1]																																					
			Byte 7								Byte 6								Byte 5								Byte 4							
			63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Checkbits [7:0]	7		*				*					*			*		*	*		*	*	*	*				*	*				*	*	
	6				*	*			*	*					*		*		*				*				*			*	*			*
	5		*	*	*	*					*				*	*		*		*	*				*	*	*				*			
	4		*		*			*		*					*		*		*			*			*	*						*		
	3										*	*	*	*	*		*				*		*				*	*	*	*					
	2						*		*					*		*			*	*				*	*							*		*
	1			*		*		*		*	*	*	*	*						*		*		*		*						*		*
	0			*			*	*	*				*			*	*	*		*				*				*		*	*		*		
		XOR sum	Byte 3								Byte 2								Byte 1								Byte 0							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7						*		*				*		*				*	*			*	*			*				*	*		*
	6		*	*		*	*				*				*				*	*		*	*					*	*				*	*
	5				*	*			*	*	*			*	*	*					*		*	*		*								
	4		*		*	*				*		*	*				*	*	*			*			*	*				*		*		
	3		*	*	*			*						*	*		*	*			*				*		*	*	*			*		
	2			*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
	1			*						*		*			*	*	*			*				*				*			*	*	*	
	0			*		*		*		*				*				*				*				*					*	*		*	*
		XOR sum	Address Bit[2]																																							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7			*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*		Unused	Unused	Unused
	6			*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
	5			*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*				
	4		*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*					
	3		*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*				
	2		*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*				
	1		*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*				
	0		*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*				

1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

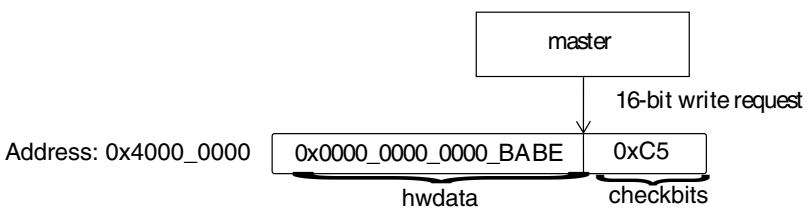
2. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

32.5.1.1 e2eECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAMC control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

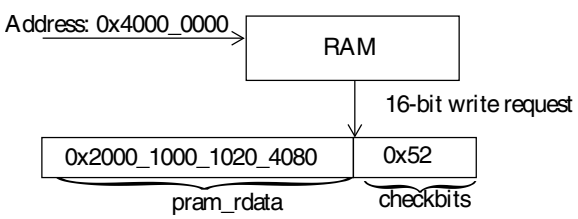
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this "zero-padded" write data.

  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

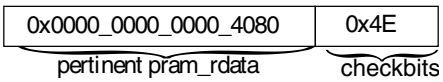


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

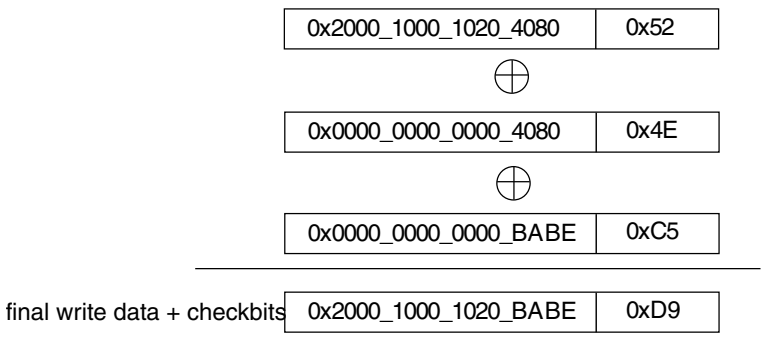
  If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

32.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the RAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions:

- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor also verifies the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

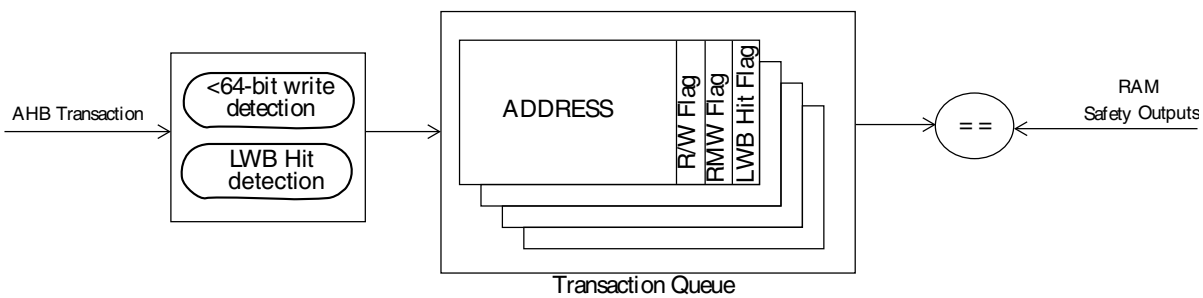


Figure 32-3. RAM controller transaction monitor block diagram

Chapter 33
Flash memory controller

33.1 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.
- It serves as the interface to the on-chip overlay RAM.
- It serves as the interface to on-chip System RAM that can be used as overlay RAM.
- It serves as the interface to the off-chip buddy device.[1]
- It serves as the interface to the External Bus Interface (EBI) memory space.

The flash memory controller supports two 64-bit AHB buses and a 256-bit read data interface from each flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the flash array page size and providing 256 bytes of total buffered storage. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

33.2 Features

The following list summarizes the key features of the flash memory controller:

1. The term "buddy device" refers to an optional companion chip with additional memory and debug features that is paired with a production chip during system development. It is not used in production devices.

- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle "buffer hit" read response.
- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.
- Configurable overlay remapping of logical flash accesses to on-chip calibration RAM, extended off-chip calibration RAM, on-chip system RAM, EBI memory
- Nexus trace stream interface supports message dumping to the on-chip overlay RAM

33.3 Block diagrams

Figure 33-1 provides a block diagram showing the flash memory controller and the attached flash array.

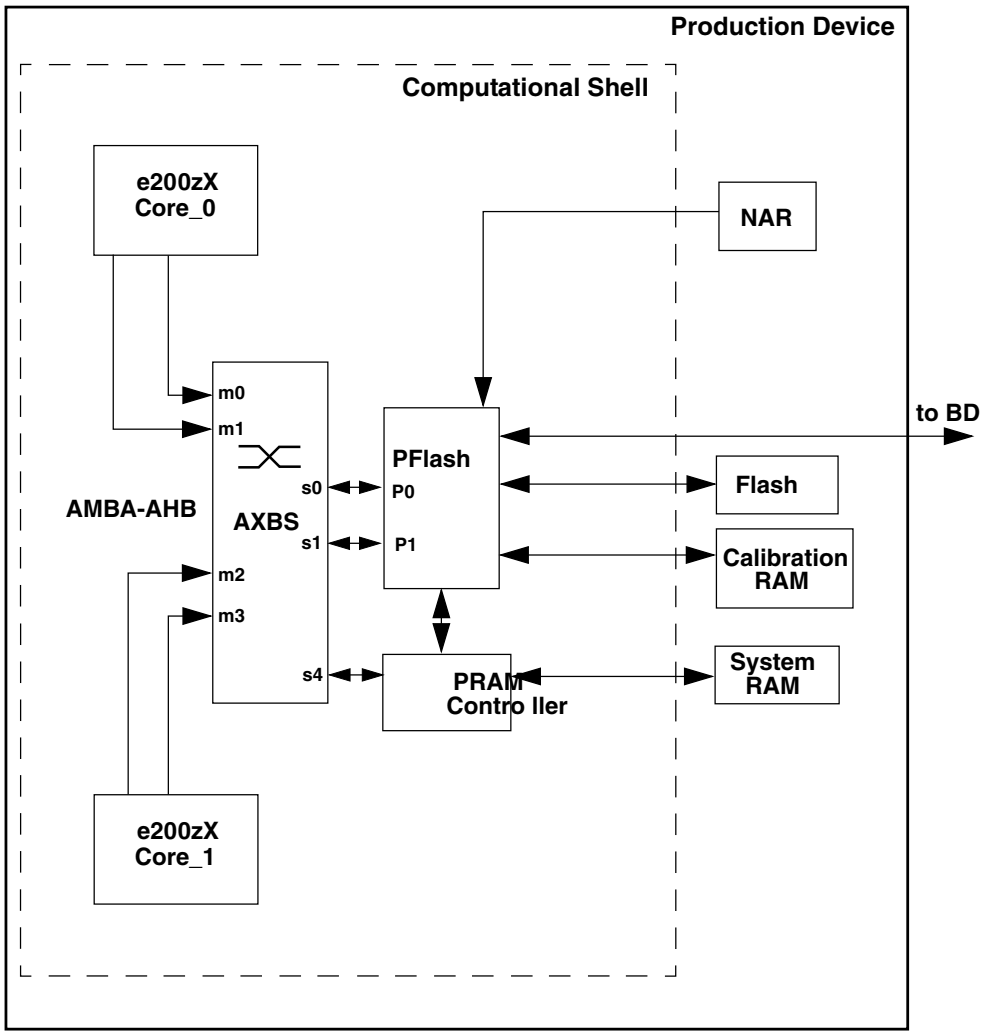


Figure 33-1. Platform-centric simple block diagram with flash memory ontroller

33.4 Flash memory controller memory map

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configurationand overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the flash memory controller module is in the midst of an operation will result in non-deterministic behavior. Software must be architected to avoid this scenario. There is no idle indicator for the flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the flash, code execution must be temporarily moved to system RAM.

PFLASH memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform Flash Configuration Register 1 (PFLASH_PFCR1)	32	R/W	0000_0601h	33.4.1/1252
4	Platform Flash Configuration Register 2 (PFLASH_PFCR2)	32	R/W	0000_0001h	33.4.2/1256
8	Platform Flash Configuration Register 3 (PFLASH_PFCR3)	32	R/W	0000_0000h	33.4.3/1259
C	Platform Flash Access Protection Register (PFLASH_PFAPR)	32	R/W	FFFF_FFFFh	33.4.4/1261
10	Platform Flash Remap Control Register (PFLASH_PFCRCR)	32	R/W	0000_0000h	33.4.5/1264
14	Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)	32	R/W	0000_0000h	33.4.6/1266
100	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD0_Word0)	32	R/W	0000_0000h	33.4.7/1271
104	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD0_Word1)	32	R/W	0000_0000h	33.4.8/1272
108	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD0_Word2)	32	R/W	0000_0000h	33.4.9/1273
110	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD1_Word0)	32	R/W	0000_0000h	33.4.7/1271
114	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD1_Word1)	32	R/W	0000_0000h	33.4.8/1272
118	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD1_Word2)	32	R/W	0000_0000h	33.4.9/1273
120	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD2_Word0)	32	R/W	0000_0000h	33.4.7/1271
124	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD2_Word1)	32	R/W	0000_0000h	33.4.8/1272
128	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD2_Word2)	32	R/W	0000_0000h	33.4.9/1273
130	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD3_Word0)	32	R/W	0000_0000h	33.4.7/1271
134	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD3_Word1)	32	R/W	0000_0000h	33.4.8/1272
138	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD3_Word2)	32	R/W	0000_0000h	33.4.9/1273
140	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD4_Word0)	32	R/W	0000_0000h	33.4.7/1271
144	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD4_Word1)	32	R/W	0000_0000h	33.4.8/1272

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
148	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD4_Word2)	32	R/W	0000_0000h	33.4.9/1273
150	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD5_Word0)	32	R/W	0000_0000h	33.4.7/1271
154	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD5_Word1)	32	R/W	0000_0000h	33.4.8/1272
158	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD5_Word2)	32	R/W	0000_0000h	33.4.9/1273
160	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD6_Word0)	32	R/W	0000_0000h	33.4.7/1271
164	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD6_Word1)	32	R/W	0000_0000h	33.4.8/1272
168	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD6_Word2)	32	R/W	0000_0000h	33.4.9/1273
170	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD7_Word0)	32	R/W	0000_0000h	33.4.7/1271
174	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD7_Word1)	32	R/W	0000_0000h	33.4.8/1272
178	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD7_Word2)	32	R/W	0000_0000h	33.4.9/1273
180	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD8_Word0)	32	R/W	0000_0000h	33.4.7/1271
184	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD8_Word1)	32	R/W	0000_0000h	33.4.8/1272
188	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD8_Word2)	32	R/W	0000_0000h	33.4.9/1273
190	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD9_Word0)	32	R/W	0000_0000h	33.4.7/1271
194	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD9_Word1)	32	R/W	0000_0000h	33.4.8/1272
198	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD9_Word2)	32	R/W	0000_0000h	33.4.9/1273
1A0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD10_Word0)	32	R/W	0000_0000h	33.4.7/1271
1A4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD10_Word1)	32	R/W	0000_0000h	33.4.8/1272
1A8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD10_Word2)	32	R/W	0000_0000h	33.4.9/1273
1B0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD11_Word0)	32	R/W	0000_0000h	33.4.7/1271
1B4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD11_Word1)	32	R/W	0000_0000h	33.4.8/1272
1B8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD11_Word2)	32	R/W	0000_0000h	33.4.9/1273

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD12_Word0)	32	R/W	0000_0000h	33.4.7/1271
1C4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD12_Word1)	32	R/W	0000_0000h	33.4.8/1272
1C8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD12_Word2)	32	R/W	0000_0000h	33.4.9/1273
1D0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD13_Word0)	32	R/W	0000_0000h	33.4.7/1271
1D4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD13_Word1)	32	R/W	0000_0000h	33.4.8/1272
1D8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD13_Word2)	32	R/W	0000_0000h	33.4.9/1273
1E0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD14_Word0)	32	R/W	0000_0000h	33.4.7/1271
1E4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD14_Word1)	32	R/W	0000_0000h	33.4.8/1272
1E8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD14_Word2)	32	R/W	0000_0000h	33.4.9/1273
1F0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD15_Word0)	32	R/W	0000_0000h	33.4.7/1271
1F4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD15_Word1)	32	R/W	0000_0000h	33.4.8/1272
1F8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD15_Word2)	32	R/W	0000_0000h	33.4.9/1273
200	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD16_Word0)	32	R/W	0000_0000h	33.4.7/1271
204	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD16_Word1)	32	R/W	0000_0000h	33.4.8/1272
208	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD16_Word2)	32	R/W	0000_0000h	33.4.9/1273
210	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD17_Word0)	32	R/W	0000_0000h	33.4.7/1271
214	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD17_Word1)	32	R/W	0000_0000h	33.4.8/1272
218	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD17_Word2)	32	R/W	0000_0000h	33.4.9/1273
220	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD18_Word0)	32	R/W	0000_0000h	33.4.7/1271
224	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD18_Word1)	32	R/W	0000_0000h	33.4.8/1272
228	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD18_Word2)	32	R/W	0000_0000h	33.4.9/1273
230	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD19_Word0)	32	R/W	0000_0000h	33.4.7/1271

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
234	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD19_Word1)	32	R/W	0000_0000h	33.4.8/1272
238	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD19_Word2)	32	R/W	0000_0000h	33.4.9/1273
240	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD20_Word0)	32	R/W	0000_0000h	33.4.7/1271
244	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD20_Word1)	32	R/W	0000_0000h	33.4.8/1272
248	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD20_Word2)	32	R/W	0000_0000h	33.4.9/1273
250	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD21_Word0)	32	R/W	0000_0000h	33.4.7/1271
254	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD21_Word1)	32	R/W	0000_0000h	33.4.8/1272
258	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD21_Word2)	32	R/W	0000_0000h	33.4.9/1273
260	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD22_Word0)	32	R/W	0000_0000h	33.4.7/1271
264	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD22_Word1)	32	R/W	0000_0000h	33.4.8/1272
268	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD22_Word2)	32	R/W	0000_0000h	33.4.9/1273
270	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD23_Word0)	32	R/W	0000_0000h	33.4.7/1271
274	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD23_Word1)	32	R/W	0000_0000h	33.4.8/1272
278	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD23_Word2)	32	R/W	0000_0000h	33.4.9/1273
280	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD24_Word0)	32	R/W	0000_0000h	33.4.7/1271
284	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD24_Word1)	32	R/W	0000_0000h	33.4.8/1272
288	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD24_Word2)	32	R/W	0000_0000h	33.4.9/1273
290	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD25_Word0)	32	R/W	0000_0000h	33.4.7/1271
294	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD25_Word1)	32	R/W	0000_0000h	33.4.8/1272
298	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD25_Word2)	32	R/W	0000_0000h	33.4.9/1273
2A0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD26_Word0)	32	R/W	0000_0000h	33.4.7/1271
2A4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD26_Word1)	32	R/W	0000_0000h	33.4.8/1272

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2A8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD26_Word2)	32	R/W	0000_0000h	33.4.9/1273
2B0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD27_Word0)	32	R/W	0000_0000h	33.4.7/1271
2B4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD27_Word1)	32	R/W	0000_0000h	33.4.8/1272
2B8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD27_Word2)	32	R/W	0000_0000h	33.4.9/1273
2C0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD28_Word0)	32	R/W	0000_0000h	33.4.7/1271
2C4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD28_Word1)	32	R/W	0000_0000h	33.4.8/1272
2C8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD28_Word2)	32	R/W	0000_0000h	33.4.9/1273
2D0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD29_Word0)	32	R/W	0000_0000h	33.4.7/1271
2D4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD29_Word1)	32	R/W	0000_0000h	33.4.8/1272
2D8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD29_Word2)	32	R/W	0000_0000h	33.4.9/1273
2E0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD30_Word0)	32	R/W	0000_0000h	33.4.7/1271
2E4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD30_Word1)	32	R/W	0000_0000h	33.4.8/1272
2E8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD30_Word2)	32	R/W	0000_0000h	33.4.9/1273
2F0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD31_Word0)	32	R/W	0000_0000h	33.4.7/1271
2F4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD31_Word1)	32	R/W	0000_0000h	33.4.8/1272
2F8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD31_Word2)	32	R/W	0000_0000h	33.4.9/1273

33.4.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)

The PFlash Configuration Register 1 (PFCR1) controls the operation of Port p0 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for information on the masters.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			RWSC					0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

PFLASH_PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	NOTE: A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup. 000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1xx Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. The required settings are documented in the device data sheet. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field applies to the configuration of Port0 and Port1. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS]=1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

33.4.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)

The PFlash Configuration Register 2 (PFCR2) controls the operation of Port1 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for details about the masters.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									P1_DPFEN	0	P1_IPFEN	0	P1_PFLIM		P1_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PFLASH_PFCR2 field descriptions

Field	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset.

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset.

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss.

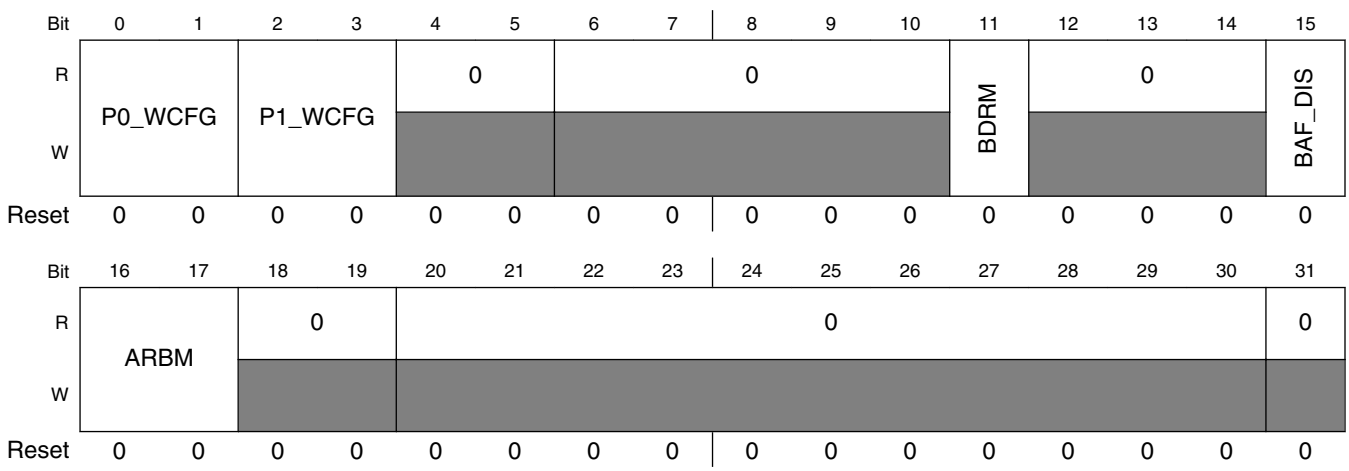
Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

33.4.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)

Address: 0h base + 8h offset = 8h



PFLASH_PFCR3 field descriptions

Field	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
2–3 P1_WCFG	Port1 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 BDRM	Buddy Device Read Mode - This field controls the PFLASH_C55FM bus response when the master presents a read request targeting the extended overlay RAM on the Buddy Device(BD) using the direct, non-remapped address. This field is ignored on production ECUs where the Buddy Device is not present. On production ECUs, this memory map region is provided for the "OLDA" (on-line data acquisition) function. Read requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address are terminated with error on the system bus. This field is ignored on write requests targeting the extended overlay RAM using the direct, non-remapped address. Write requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address are ignored and terminate cleanly on the system bus. Write requests on engineering ECUs targeting the extended overlay RAM using the direct, non-remapped address are executed. This field is initialized by hardware reset. 0 Read requests targeting the extended overlay RAM on the Buddy Device(BD) using the direct, non-remapped address are aborted, and the request is terminated with error on the system bus. 1 Read requests targeting the extended overlay RAM on the Buddy Device(BD) using the direct, non-remapped address are allowed. The PFLASH_C55FM returns the requested data from the extended overlay RAM, which resides on the BD. Any system bus errors returned by the BD are passed onto the master.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 BAF_DIS	BAF Disable. This field controls executable access to the BAF (Boot Assist Flash) region of the flash. Once this field is set, attempted instruction accesses targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through both, Port0 and Port1. Data-type accesses to the BAF region are not affected by this field.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF flash region is allowed. 1 Executable access to the BAF flash region is prohibited.
16–17 ARBM	Arbitration Mode. This 2-bit field controls the arbitration of concurrent flash access requests from the two AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 10 Round-robin arbitration 11 Round-robin arbitration
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

33.4.4 Platform Flash Access Protection Register (PFLASH_PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

NOTE

See the chip-specific platform flash controller information for details about the actual masters available on the device.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PFLASH_PFAPR field descriptions

Field	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

33.4.5 Platform Flash Remap Control Register (PFLASH_PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

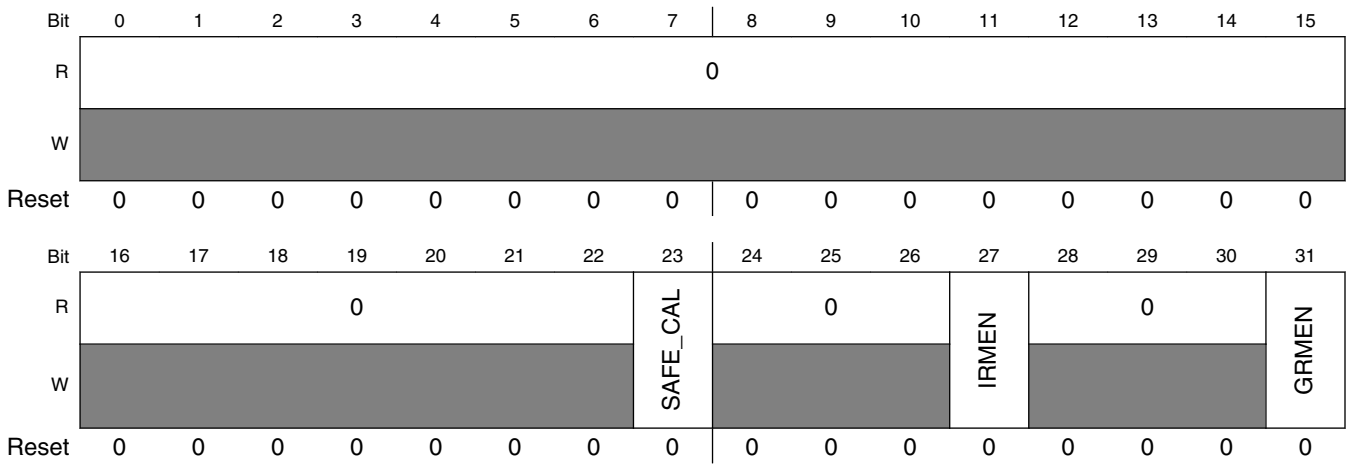
CAUTION

Overlay remap functions, when used with line read buffers enabled (PFLASH_PFCRx[BFy_EN] field(s)='1') have a potential to cause data coherency issues, i.e., prefetched data in a line read buffer can become out-of-sync with data contained at the associated flash address range. To prevent this issue, always clear the flash controller line read buffers prior to enabling any remap region.

To clear the flash controller line read buffers, issue an interlock write command to a valid flash memory address. There will not be an actual flash program operation performed, but the flash controller mini-cache will be cleared as a side effect of executing the command. The interlock write must be issued from a core having appropriate access to flash. The sequence is as follows:

1. Set the C55FMC_MCR[PGM] field to '1'
2. Issue a 64-bit write to a valid flash address on a 64-bit boundary, i.e., the least significant 5 bits of the address are '0'
3. Clear C55FMC_MCR[PGM]

Address: 0h base + 10h offset = 10h



PFLASH_PFCRCR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 32 to 16. This field is initialized by hardware reset. 1 Established calibration overlay regions are considered safety-critical. 0 Established calibration overlay regions are not considered safety-critical.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remapping evaluation on system flash accesses. This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

33.4.6 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 32 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	CRD8EN	CRD9EN	CRD10EN	CRD11EN	CRD12EN	CRD13EN	CRD14EN	CRD15EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRD16EN	CRD17EN	CRD18EN	CRD19EN	CRD20EN	CRD21EN	CRD22EN	CRD23EN	CRD24EN	CRD25EN	CRD26EN	CRD27EN	CRD28EN	CRD29EN	CRD30EN	CRD31EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 0 invalid 1 Calibration Remap Descriptor 0 valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 1 invalid 1 Calibration Remap Descriptor 1 valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 2 invalid 1 Calibration Remap Descriptor 2 valid
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 3 invalid 1 Calibration Remap Descriptor 3 valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 4 invalid 1 Calibration Remap Descriptor 4 valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 5 invalid 1 Calibration Remap Descriptor 5 valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 6 invalid 1 Calibration Remap Descriptor 6 valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 7 invalid 1 Calibration Remap Descriptor 7 valid
8 CRD8EN	Calibration Remap Descriptor 8 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 8 invalid 1 Calibration Remap Descriptor 8 valid
9 CRD9EN	Calibration Remap Descriptor 9 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 9 invalid 1 Calibration Remap Descriptor 9 valid
10 CRD10EN	Calibration Remap Descriptor 10 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 10 invalid 1 Calibration Remap Descriptor 10 valid
11 CRD11EN	Calibration Remap Descriptor 11 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 11 invalid 1 Calibration Remap Descriptor 11 valid
12 CRD12EN	Calibration Remap Descriptor 12 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 12 invalid 1 Calibration Remap Descriptor 12 valid
13 CRD13EN	Calibration Remap Descriptor 13 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 13 invalid 1 Calibration Remap Descriptor 13 valid
14 CRD14EN	Calibration Remap Descriptor 14 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 14 invalid 1 Calibration Remap Descriptor 14 valid
15 CRD15EN	Calibration Remap Descriptor 15 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 15 invalid 1 Calibration Remap Descriptor 15 valid
16 CRD16EN	Calibration Remap Descriptor 16 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 16 invalid 1 Calibration Remap Descriptor 16 valid
17 CRD17EN	Calibration Remap Descriptor 17 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 17 invalid 1 Calibration Remap Descriptor 17 valid
18 CRD18EN	Calibration Remap Descriptor 18 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 18 invalid 1 Calibration Remap Descriptor 18 valid
19 CRD19EN	Calibration Remap Descriptor 19 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 19 invalid 1 Calibration Remap Descriptor 19 valid
20 CRD20EN	Calibration Remap Descriptor 20 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 20 invalid 1 Calibration Remap Descriptor 20 valid
21 CRD21EN	Calibration Remap Descriptor 21 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 21 invalid 1 Calibration Remap Descriptor 21 valid
22 CRD22EN	Calibration Remap Descriptor 22 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 22n invalid 1 Calibration Remap Descriptor 22 valid
23 CRD23EN	Calibration Remap Descriptor 23 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 23 invalid 1 Calibration Remap Descriptor 23 valid
24 CRD24EN	Calibration Remap Descriptor 24 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 24 invalid 1 Calibration Remap Descriptor 24 valid
25 CRD25EN	Calibration Remap Descriptor 25 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 25 invalid 1 Calibration Remap Descriptor 25 valid
26 CRD26EN	Calibration Remap Descriptor 26 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 26 invalid 1 Calibration Remap Descriptor 26 valid
27 CRD27EN	Calibration Remap Descriptor 27 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 27 invalid 1 Calibration Remap Descriptor 27 valid
28 CRD28EN	Calibration Remap Descriptor 28 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 28 invalid 1 Calibration Remap Descriptor 28 valid
29 CRD29EN	Calibration Remap Descriptor 29 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 29 invalid 1 Calibration Remap Descriptor 29 valid
30 CRD30EN	Calibration Remap Descriptor 30 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 30 invalid 1 Calibration Remap Descriptor 30 valid
31 CRD31EN	Calibration Remap Descriptor 31 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 31 invalid 1 Calibration Remap Descriptor 31 valid

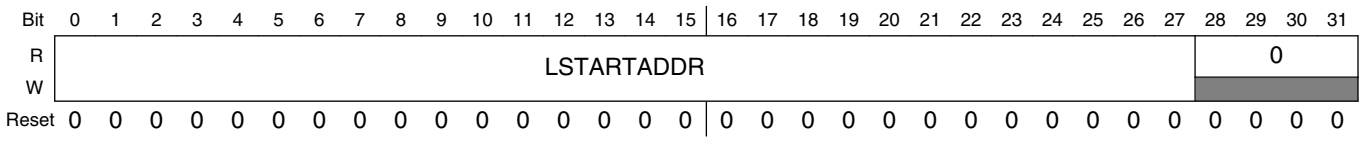
33.4.7 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-size logical start (byte) address of the calibration remap region. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

The hardware performs basic checks on the validity of the logical start address when this word is written; if a non-supported logical start address is defined, the register write is terminated with an error and the register left unchanged and the valid bit cleared. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 100h offset + (16d × i), where i=0d to 31d



PFLASH_PFCRD*n*_Word0 field descriptions

Field	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

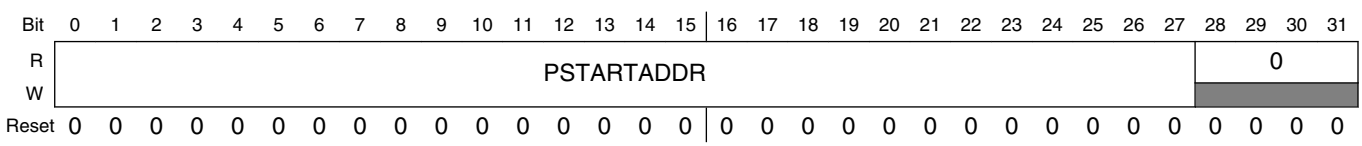
33.4.8 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-size byte physical start (byte) address of the calibration remap region.

The contents of this word define the targeted destination overlay memory. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Attempted writes to this register with illegal values, as defined by the sizes and locations of the overlay memories, are terminated with an error and clear the valid bit. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 104h offset + (16d × i), where i=0d to 31d



PFLASH_PFCRD*n*_Word1 field descriptions

Field	Description
0–27 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to

Table continues on the next page...

PFLASH_PFCRD*n*_Word1 field descriptions (continued)

Field	Description
	the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

33.4.9 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

Address: 0h base + 108h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word2 field descriptions

Field	Description
0 M0EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 0 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 0 as defined by this region descriptor.
1 M1EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 1 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 1 as defined by this region descriptor.
2 M2EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 2 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 2 as defined by this region descriptor.
3 M3EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 3 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 3 as defined by this region descriptor.
4 M4EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 4 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 4 as defined by this region descriptor.
5 M5EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 5 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 5 as defined by this region descriptor.
6 M6EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 6 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 6 as defined by this region descriptor.
7 M7EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 7 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 7 as defined by this region descriptor.
8 M8EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 8 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 8 as defined by this region descriptor.
9 M9EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 9 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 9 as defined by this region descriptor.
10 M10EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 10 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 10 as defined by this region descriptor.
11 M11EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 11 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 11 as defined by this region descriptor.
12 M12EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 12 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 12 as defined by this region descriptor.
13 M13EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 13 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 13 as defined by this region descriptor.
14 M14EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 14 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 14 as defined by this region descriptor.
15 M15EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 15 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 15 as defined by this region descriptor.
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CRDSize	Calibration Remap Descriptor *n* Size - This field specifies the size of the calibration remap region. 00000 Reserved 00001 Reserved 00010 Reserved 00011 Reserved

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	00100 Reserved 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero 01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero 01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero 01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero 01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero 10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero 10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero 10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero 10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero 10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero 10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero 10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero 10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

33.5 Functional description

The flash memory controller interfaces between:

- The AHB system bus ports
- Nexus trace
- Flash memory array
- Overlay RAM
- Off-chip Buddy Device
- System RAM
- EBI memory

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it

onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

33.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

33.5.2 Access protections

33.5.2.1 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in Platform Flash Access Protection Register (PFLASH_PFAPR). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

33.5.2.2 BAF execution disable

The flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code. Setting the BAF Disable field (BAF_DIS) in the PFlash Configuration Register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on Port0 or Port1 are evaluated. If

the flash memory controller detects an attempted instruction fetch access to the BAF region of the flash, the flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

33.5.3 Read cycles - buffer miss

On an incoming AHB read request, a mini-cache lookup and access privilege evaluation are performed during the AHB address phase. In the event of a buffer miss, a flash access is initiated. If globally enabled, the calibration remap evaluation is also performed. If the access is to be remapped, the appropriate access is routed to the destination memory.

If the flash access was the direct result of an AHB transaction, the corresponding page buffer is loaded and marked as most-recently-used. If the flash access was the result of a speculative prefetch to the next sequential line, it is loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs as a result of an AHB read request.

33.5.4 Read cycles - buffer hit

Single cycle read responses to the AHB are possible with the flash memory controller when the requested read access was previously loaded into one of the page buffers. In these "buffer hit" cases, read data is returned on the system bus with a zero wait-state response.

33.5.5 Error termination

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding Read Access Control or Write Access Control settings do not allow the access, thus causing a protection violation. See Platform Flash Access Protection Register (PFLASH_PFAPR) for more detail. In this case the flash memory controller does not initiate a flash array access.
- Attempted instruction fetch by an AHB master to the BAF flash region when the PFCR3[BAF_DISABLE] field in Platform Flash Configuration Register 3 (PFCR3) is set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed. See Platform Flash Configuration Register 3

(PFLASH_PFCR3) for more detail. In this case, the flash memory controller does not initiate a flash array access.
- Attempted access by an AHB master to a flash region, where the corresponding censorship control is asserted.
- The flash or overlay RAM returns an error response on an attempted access to a partition that is unavailable. (see Flash error response operation or Embedded Flash Memory).
- Attempted access by an AHB master to a reserved region in the flash memory map.

33.5.6 Flash error response operation

The flash array may signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received the flash memory controller does not update or validate a page read buffer. An error response may be signaled on read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash memory chapter.

33.5.7 Censorship

The flash space is defined by regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation)
- Secure code flash (instruction for security module)
- Secure data flash (data for security module)
- UTEST flash
- Security flash blocks

Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

33.5.8 Security module exclusive control

During "Production at OEM" lifecycle state and beyond, the HSM can establish the secure flash blocks to be exclusive to the HSM and route program or erasing of secure flash blocks to the alternate programming interface.

The secure flash space is independently controlled for each of the following regions:

- 16 KB secure code flash block at address range 0x0060_C0000 - 0x0060_FFFF
- 64 KB secure code flash blockA at address range 0x0061_0000 - 0x0061_FFFF
- 64 KB secure code flash blockB at address range 0x0062_0000 - 0x0062_FFFF
- Two 16 KB secure data flash blocks at address range 0x0068_0000 - 0x0068_7FFF

The HSM provides flags to individually control the HSM-exclusivity for each of the secure flash regions. When a given secure flash block is marked as exclusive to the HSM, the HSM is the only system bus master granted access to that block and access attempts by non-HSM masters are terminated with error. Once HSM-exclusivity is established for a given secure flash block, the HSM is guaranteed access to that block, regardless of the state of the censorship controls. Secure flash exclusivity applies to read and write accesses.

33.5.9 Security Module Alternate Programming Interface

The secure flash space is independently controlled for each of the following regions:

- 16 KB secure code flash block at address range 0x0060_C0000 - 0x0060_FFFF
- 64 KB secure code flash blockA at address range 0x0061_0000 - 0x0061_FFFF
- 64 KB secure code flash blockB at address range 0x0062_0000 - 0x0062_FFFF
- Two 16 KB secure data flash blocks at address range 0x0068_0000 - 0x0068_7FFF

The HSM provides flags to individually control the routing interlock write for each of the four secure flash regions to either the main program/erase interface or the alternate program/erase interface. When a given secure flash block is assigned to the alternate program interface, the flash controller appropriately routes incoming interlock write requests within the given secure flash block address range to the flash's alternate program interface. Alternate program control can be controlled independent of HSM exclusivity for a given secure flash block

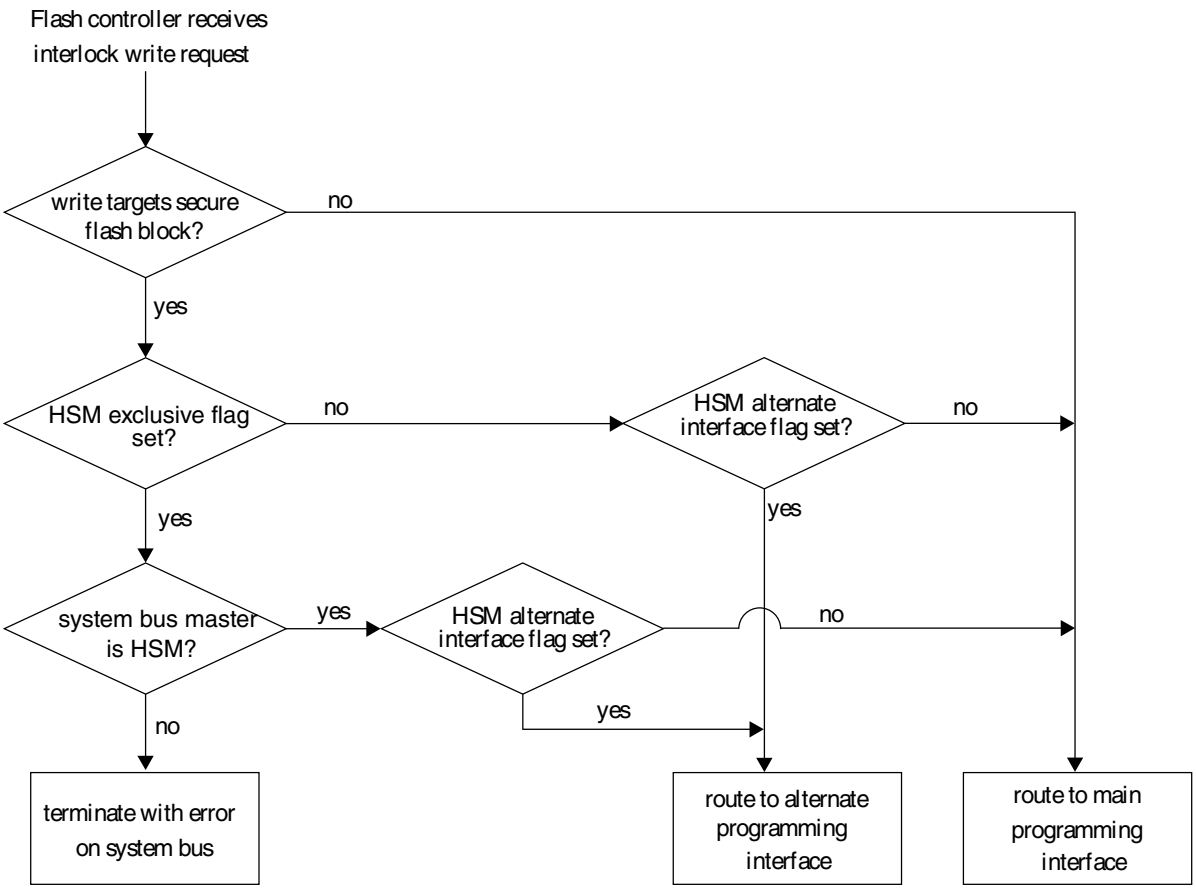


Figure 33-2. Interlock Write Alternate Programming Interface Routing Flowchart

33.5.10 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

NOTE

Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

33.5.11 Line read buffers and prefetch operation

The AHB ports of the flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see Platform Flash Configuration Register 1 (PFLASH_PFCR1)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

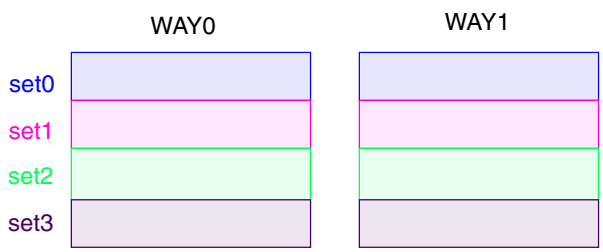


Figure 33-3. Flash memory controller 4-entry, 2-way mini-cache organization

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Platform Flash Configuration Register 1 (PFLASH_PFCR1). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2}[P{0,1}_BFEN] must be set to ‘1’,
2. PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and
3. Either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted.

Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

33.5.12 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to Platform Flash Configuration Register 2 (PFLASH_PFCR2) for a description of these controls. Prefetches are never triggered by write cycles.

33.5.13 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

33.5.14 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to Platform Flash Configuration Register 3 (PFLASH_PFCR3). The buffers can be organized as a "pool" of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

33.5.15 EBI memory support

The PFLASH_C55FM supports memory accesses via the External Bus Interface (EBI). The EBI memory can serves as an extension of flash memory space with potential to be overlaid, or the EBI memory can serve as an overlay target for remapping flash accesses. Contents fetched from the EBI memory space can be stored locally in the PFLASH_C55FM mini-cache to facilitate zero-wait state subsequent accesses.

33.5.16 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM on-chip System RAM that can be used as overlay RAM.

Recall this family of devices includes a comprehensive set of features including dedicated overlay RAM arrays to aid calibration and debug. The overlay function supports the following features:

- Can be mapped over internal flash memory
  - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Can be used to hold device debug trace stream
  - Allows limited trace based debug without requirement for complex debug tool hardware and access to high speed trace port
- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can achieve the same timing as accesses to internal flash
  - Timing for calibration accesses to a particular overlay RAM is only guaranteed if it is not being used as a destination for trace streaming at the same time
  - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function :
  - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
  - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay targets:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is only available on special ED devices.
- A portion of the system RAM can be used for overlay.

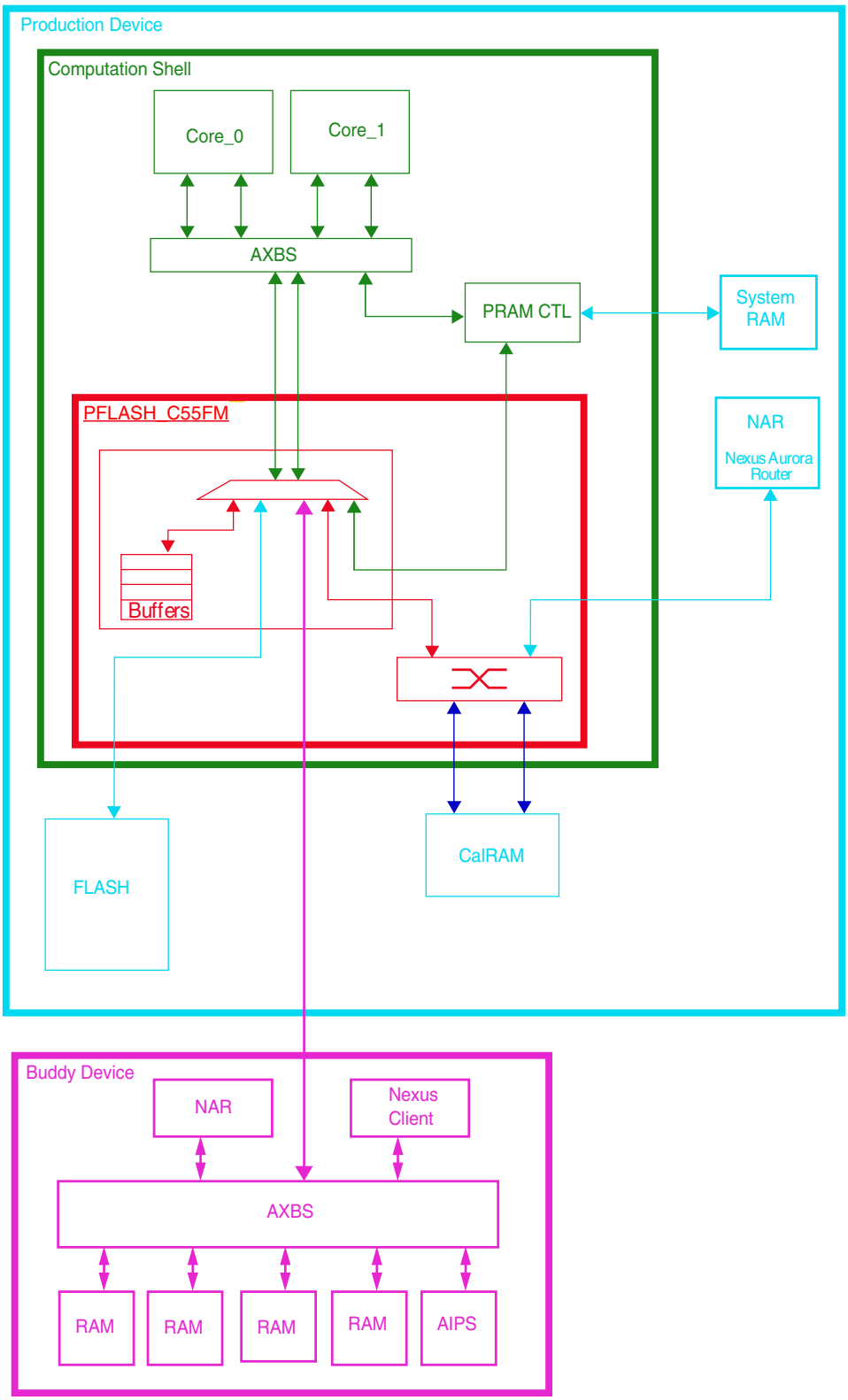


Figure 33-4. Flash memory controller and associated calibration remap targets

33.5.16.1 PFlash calibration remap to RAM

The overlay RAM is selected based on the translated physical address.

To better understand the functionality of the calibration remap, consider more detailed block diagrams, one showing the overall structure of the calibration remap function and another showing more detail on the actual remap logic. Recall the flash memory controller implements a hardware interconnection matrix connecting the three input ports and four destination memories. See Figure 33-5.

The following figure illustrates the calibration remap connection matrix.

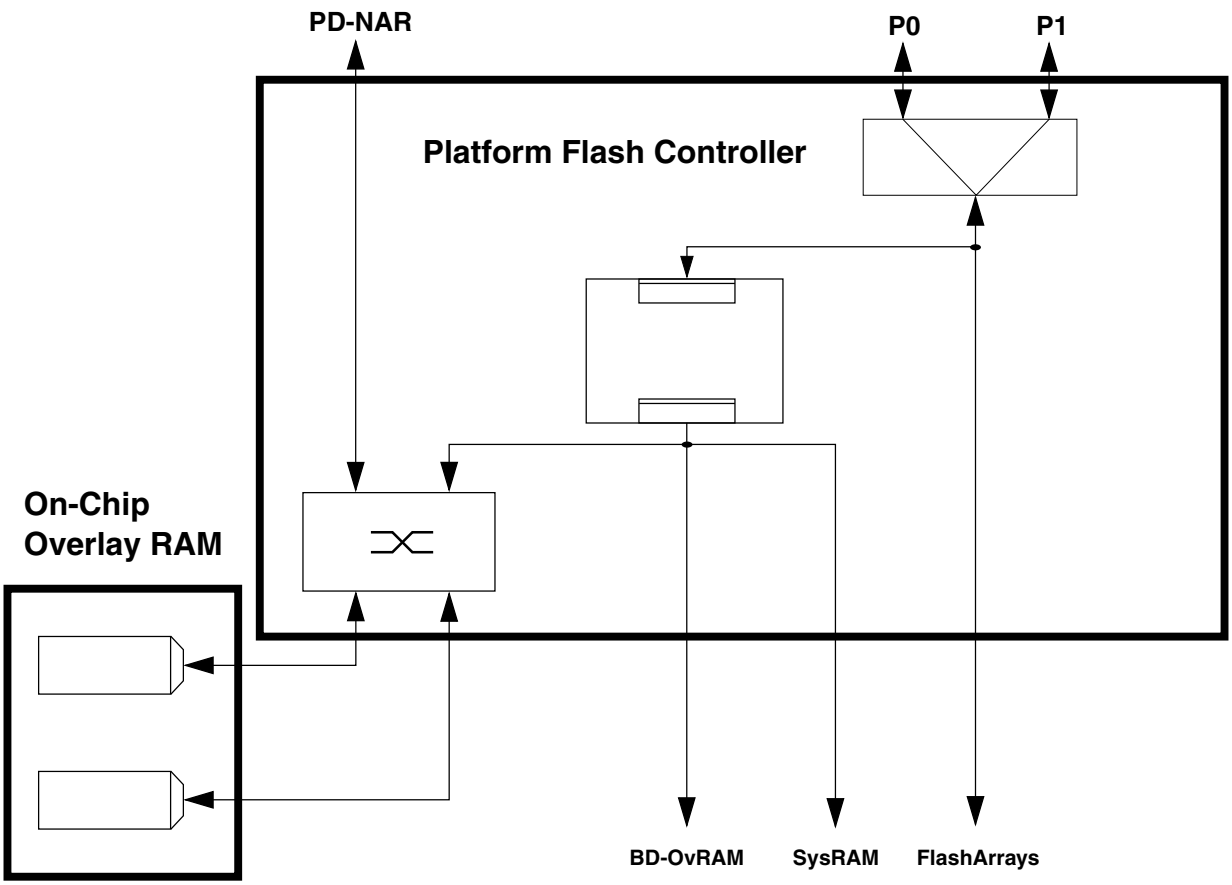


Figure 33-5. PFlash calibration remap connection matrix

Note the 64-bit on-chip overlay RAM is split into 2 equal-sized memories to allow concurrent accesses for calibration data from the P0, P1 (where these are the crossbar slave input ports) plus trace data writes from the Production Device Nexus Aurora Router (PD-NAR).

The translated physical address defines the destination memory. Table 33-1 shows the base addresses for the calibration data memories as defined by the system memory map.

Table 33-1. Calibration data memory base addresses

Calibration Memory	Base Address	Possible Access Sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1
On-chip Overlay RAM	0x0D00_----	P0, P1, PD-NAR
Buddy Device (Extended) Overlay RAM	0x0C{0,1}-_----	P0, P1

Note the trace data from the Nexus Aurora Router (PD-NAR) can only be routed to the on-chip overlay RAM; there is a different connection path in the Buddy Device from the BD-NAR into the extended overlay RAM.

The system RAM connection from the calibration remap logic is implemented via a private 64-bit connection to the platform RAM controller.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM, on-chip or buddy device (extended) overlay RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a translated physical address or, if provided by the requesting master, the physical, non-remapped system address. Byte-, halfword-, word-, and doubleword-sized data store references to the on-chip overlay RAM are supported. Byte-, halfword-, word-, and doubleword-sized store references to the buddy device (extended) overlay RAM are supported.

The read data interface of system RAM, on-chip overlay RAM, and buddy (extended) device is 64 bits wide for each, whereas the flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device. The access time when fetching data from the buddy device presents the longest latency. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

$$(\mathrm{RWSC}+2) \geq \left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}+2\right) \geq (\mathrm{OverlayRAMTransferBurstLatency})$$

Equation 20. Access time

Notice the overall latency on a random, initial flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic flash access time. This cycle accounts for pipeline stalls incurred by the flash controller state machine. The formula above is based on specified intrinsic flash access times of the C55FMC array at 150 °C. Calibration data that is loaded in the flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRD*n*) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the flash controller is presented with a flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping. Generally, not all areas of the flash memory map are available for calibration remapping. Secure flash blocks, EEPROM, BAF and UTEST are usually not available for calibration remapping. Flash access requests which present an address targeting the secure flash blocks, EEPROM, BAF or UTEST will bypass calibration remap evaluation and return flash data.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries. See Figure 33-6.

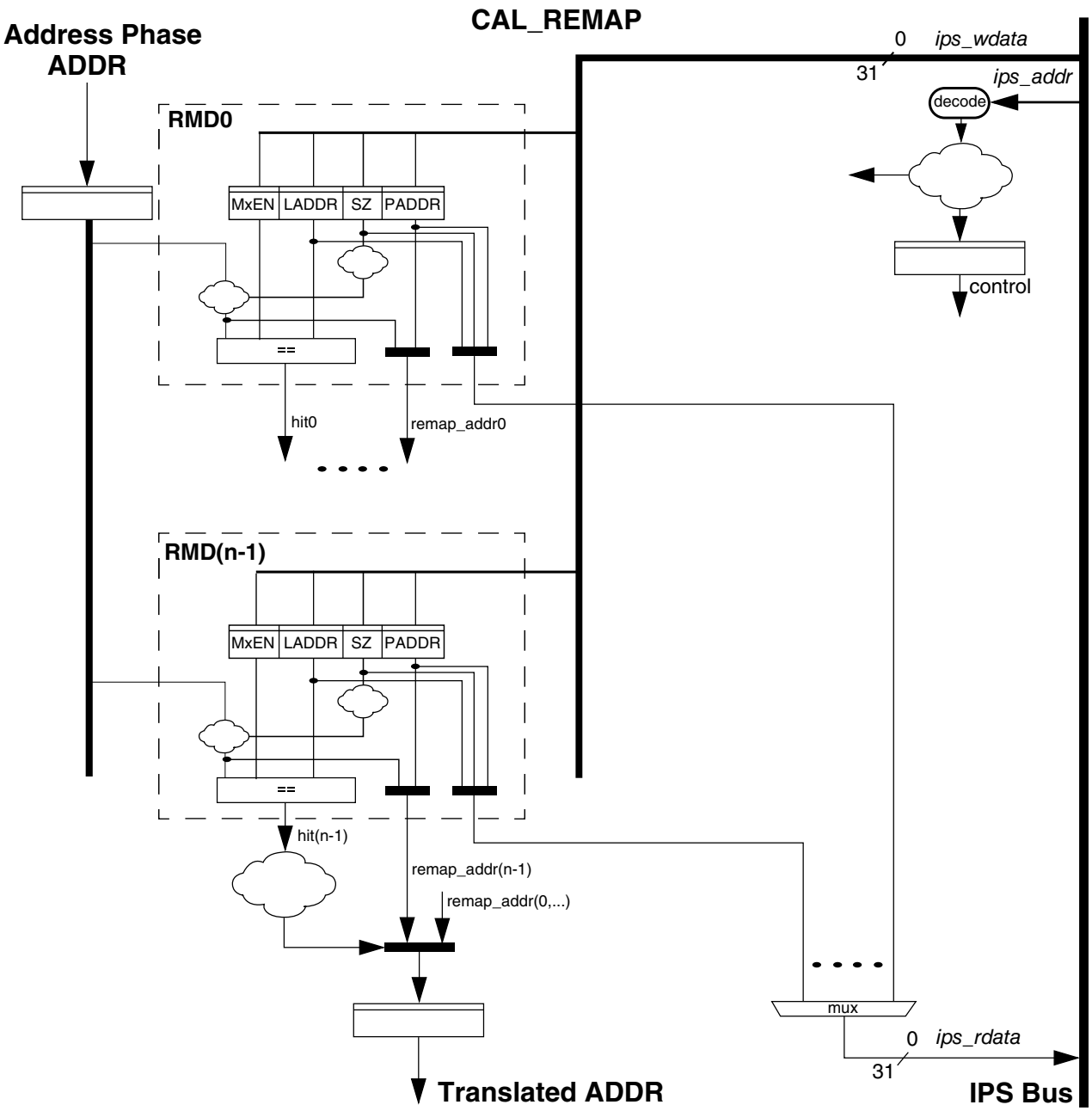


Figure 33-6. PFlash calibration remap logic detail

Each instantiation of the basic calibration remap function {RMD0,…, RMD(*n*-1)} in Figure 33-6 includes the three descriptor registers containing the logical start address (LADDR in the figure), the physical start address (PADDR), region size (SZ), and per-master enable bits (MxEN). The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap descriptor to determine the region "hit". The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration

remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region "hit" data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

33.5.17 Reliability considerations

This section summarizes mechanism to enhance the reliability of flash by correcting and detecting faults.

33.5.17.1 e2eECC End-to-End ECC

33.5.17.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all-ones codeword, since it reflects the state of an erased flash location, is treated as a valid error free encoding; in particular, it is required that the checkbits associated with the all all-ones value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all-ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all-ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

   addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

   addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3

   Before performing the interlock write to the flash array, the flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the flash array is:

   fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8

3. Read address 0x00345678. The flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

   addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

33.5.17.2 Flash address generation check

Fault detection coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Fault detection coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

33.5.17.2.1 On-Chip Overlay RAM feedback check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

33.5.17.2.2 Reliability considerations on overlay accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical flash address contribution. This is required to satisfy the e2eECC check at the originating master.

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming PFCRCR[SAFE_CAL]=1, the flash memory controller treats the set of available calibration region descriptors as a replicated pair, with each set containing half the total number of available calibration region descriptors.

If there are 32 calibration region descriptors, CRD0 – CRD31, and PFCRCR[SAFE_CAL]=1, the flash memory controller treats CRD0 – CRD15 and CRD16 – 32 as replicated sets of 16 calibration region descriptors, where:

- CRD0 and CRD16 are configured identically
- CRD1 and CRD17 are configured identically
- CRD2 and CRD18 are configured identically
- ...
- and CRD15 and CRD31 are configured identically

Expressed more generally, the contents of CRD[n] must be identical to the contents of CRD[(N/2) + n], where N is the total number of region descriptors and $0 \le n \le (N/2) - 1$.

Note

It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2) – 1] and the calibration regions defined in CRD[N/2] to CRD[N – 1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

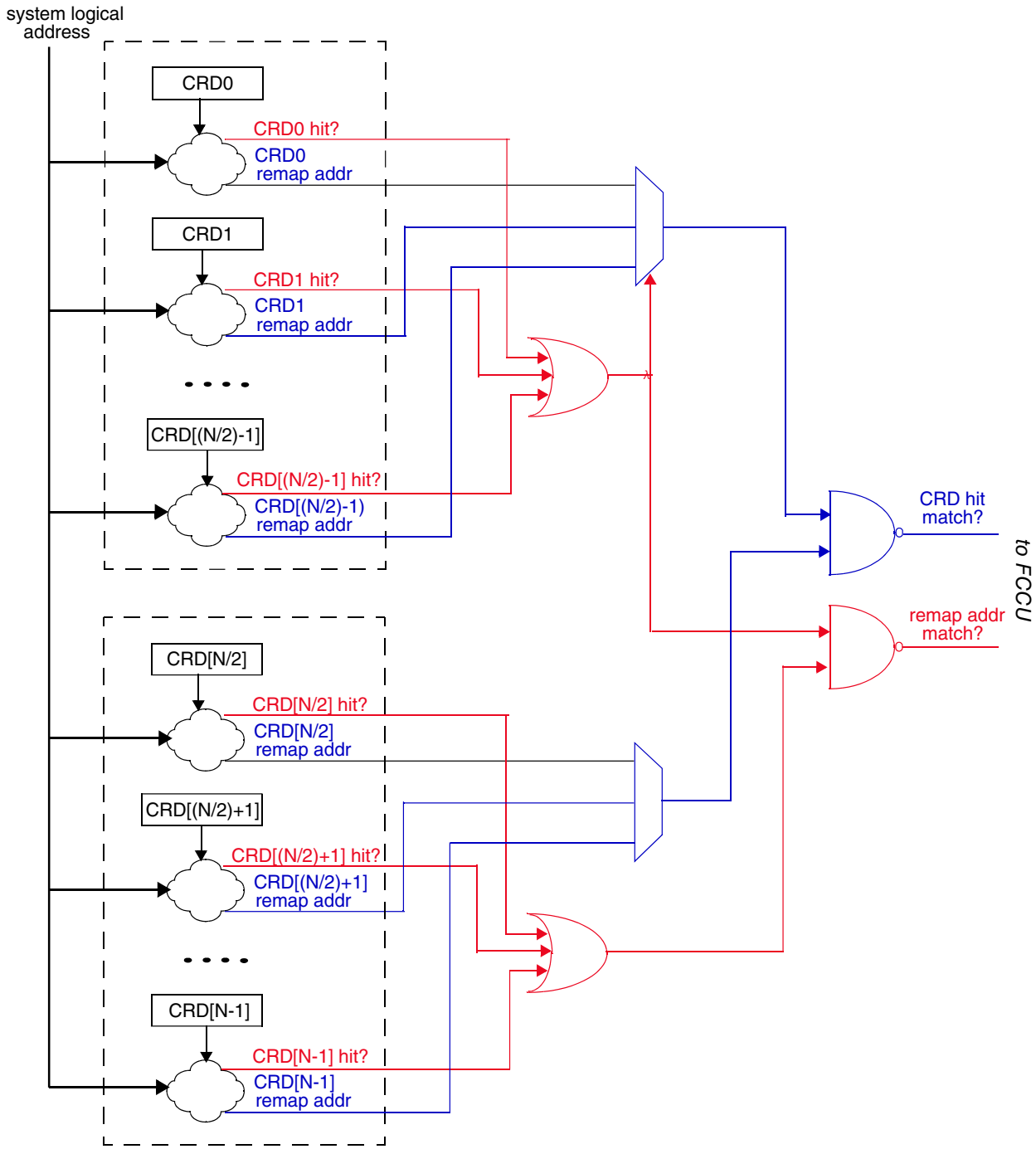


Figure 33-7. Safety-critical calibration remap datapath with redundancy

33.5.18 ECC on data flash accesses

ECC events detected on accesses to data flash blocks are suppressed from being reported to the Memory Error Management Unit (MEMU).

In the event of a single-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address, and the non-correctable error event is suppressed from being reported to the MEMU.

NOTE

EEPROM should be avoided for storage of executable code.

33.5.19 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error. More specifically, it aborts the incoming flash access requests and terminates the system bus transfer with an error.

Chapter 34
Embedded Flash Memory (c55fmc)

34.1 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

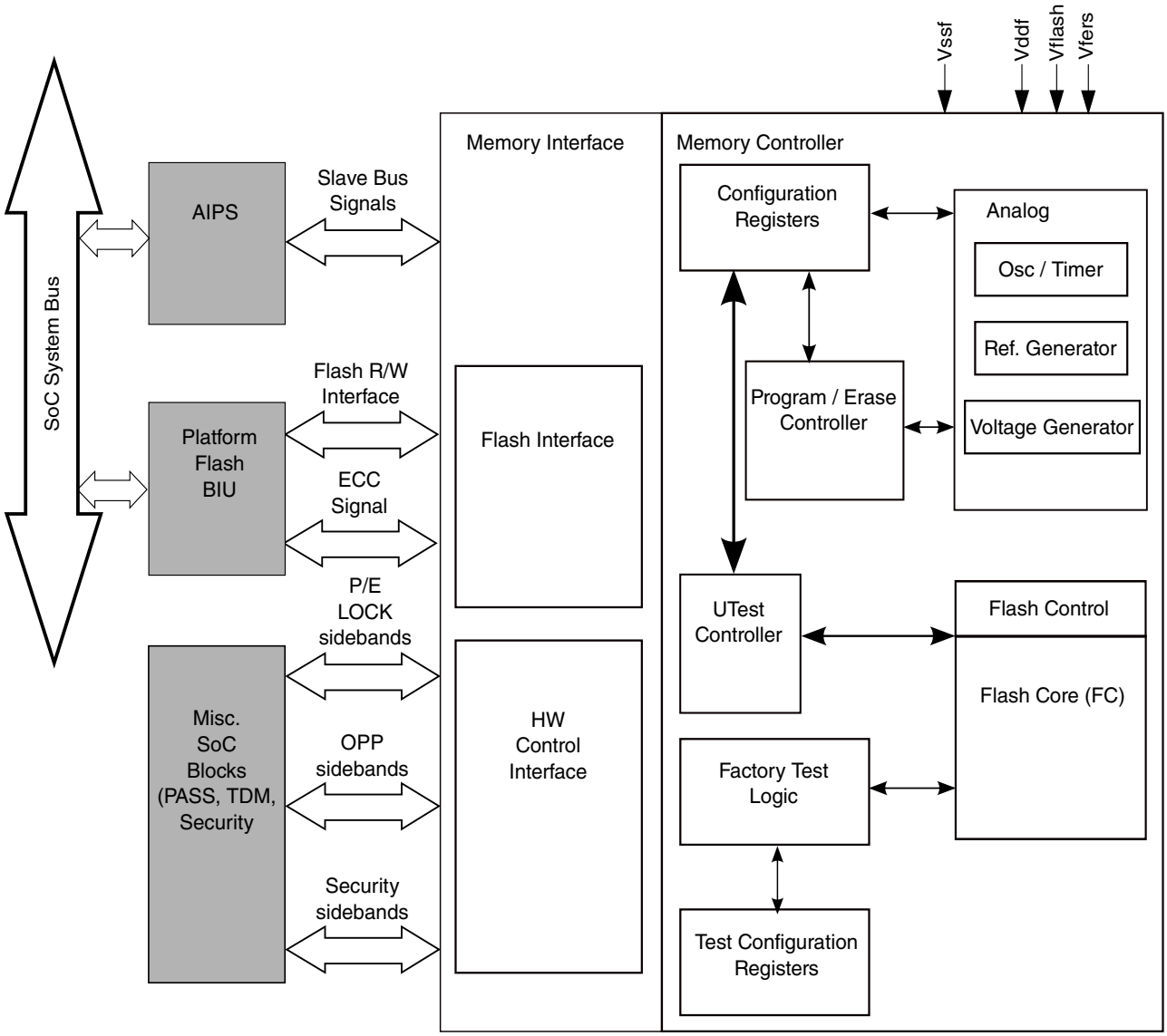


Figure 34-1. c55fmc block diagram

34.1.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see Functional Description.

34.1.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control

34.1.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.
- *Low power mode* turns off all DC current sources within the embedded flash memory on the Vflash domain, and enables VDDF to be power gated. The embedded flash memory is not accessible for read, write, program, or erase when in low power mode.
- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

34.2 UTest NVM block

The UTest NVM block may be enabled by the BIU. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block (assuming Test Mode Disable Seal is written) - thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

34.3 Test mode disable seal

Also included in the UTest NVM block is a mechanism to disable factory entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method will not be able to have possible failures analyzed by factory failure analysts. Once sealed, the Utest block becomes OTP (Erase Locked, and Over Program Protection enabled). The Utest Block also is not accessible in Test Mode.

Protection of this sort prevents all high-voltage operations to the flash executed by the internal state-machine, as well as reads through this state machine, and reads through the Array Integrity state machine when using Test mode interfaces.

Utest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and Single Bit Corrections and Double Bit Detections will be logged. Protection through other interfaces is protected by normal User mode protection mechanisms, and are device-specific.

The method to disable factory entry into Test mode is to first program the Test Mode Disable Seal location to be 0x2D3C_4B5A . Once the next reset is asserted, Test mode will be disabled.

It is possible to create a password to enable factory entry into test mode. This can be programmed into the Test Mode Disable Override Passcode. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and will not be accepted to override. If it is desired by the customer that override never be possible, one of the three invalid passcodes should be put into this location. Passcode may be entered to authenticate entry (if enabled) by doing a 32-bit register write to register address 0x90. The UTest NVM block is always protected even if the Test Mode Disable Seal password is written. UTest operations Margin Read and Array Integrity are always protected even if the Test Mode Disable Seal password is written.

Only blocks selected in the Test Mode Disable Block Select field are controlled by the Test Mode Disable feature. Thus it is possible for customers to selectively pick blocks that have this type of protection, and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test Mode Disable Block Select field(s) will designate blocks that are controlled by the Test Mode Disable feature. If either Test Mode Disable Block Select - 0 or Test Mode Disable Block Select - 1 have a 0 programmed, that block will be protected. In order to let two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1. The bit of these two regions are logically ANDed, to define which blocks are effectively selected for the Test Mode Disable feature. The Block Select field is organized as follows, and aligns with how blocks are defined in the LOCK registers.

Table 34-1. Test mode disable block select

Block	Bits used to select blocks
Low Block Disable	Data[13:0]
Unused	Data[15:14]
Mid Block Disable	Data[31:16]
High Block Disable	Data[47:32]
256K Block Disable	Data[95:48]

34.4 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks..

C55FMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (C55FMC_MCR)	32	R/W	0000_0600h	34.4.1/1306
4	Alternate Module Configuration Register (C55FMC_MCRA)	32	R/W	See section	34.4.2/1312
8	Extended Module Configuration Register (C55FMC_MCRE)	32	R	See section	34.4.3/1314
10	Lock 0 register (C55FMC_LOCK0)	32	R/W	BFFF_FFFFh	34.4.4/1318
14	Lock 1 register (C55FMC_LOCK1)	32	R/W	0000_FFFFh	34.4.5/1320
18	Lock 2 register (C55FMC_LOCK2)	32	R/W	FFFF_FFFFh	34.4.6/1321
1C	Lock 3 register (C55FMC_LOCK3)	32	R/W	0000_FFFFh	34.4.7/1321
28	Alternate Lock 0 register (C55FMC_LOCK0A)	32	R/W	BFFF_FFFFh	34.4.8/1322
2C	Alternate Lock 1 register (C55FMC_LOCK1A)	32	R/W	0000_FFFFh	34.4.9/1324
38	Select 0 register (C55FMC_SEL0)	32	R/W	0000_0000h	34.4.10/ 1325
3C	Select 1 register (C55FMC_SEL1)	32	R/W	0000_0000h	34.4.11/ 1326
40	Select 2 register (C55FMC_SEL2)	32	R/W	0000_0000h	34.4.12/ 1327
44	Select 3 register (C55FMC_SEL3)	32	R/W	0000_0000h	34.4.13/ 1328
50	Address register (C55FMC_ADR)	32	R	0000_0000h	34.4.14/ 1329
54	UTest 0 register (C55FMC_UT0)	32	R/W	0000_0001h	34.4.15/ 1331
58	UMISR register (C55FMC_UM0)	32	R/W	0000_0000h	34.4.16/ 1334
5C	UMISR register (C55FMC_UM1)	32	R/W	0000_0000h	34.4.16/ 1334
60	UMISR register (C55FMC_UM2)	32	R/W	0000_0000h	34.4.16/ 1334
64	UMISR register (C55FMC_UM3)	32	R/W	0000_0000h	34.4.16/ 1334
68	UMISR register (C55FMC_UM4)	32	R/W	0000_0000h	34.4.16/ 1334
6C	UMISR register (C55FMC_UM5)	32	R/W	0000_0000h	34.4.16/ 1334
70	UMISR register (C55FMC_UM6)	32	R/W	0000_0000h	34.4.16/ 1334
74	UMISR register (C55FMC_UM7)	32	R/W	0000_0000h	34.4.16/ 1334
78	UMISR register (C55FMC_UM8)	32	R/W	0000_0000h	34.4.16/ 1334
7C	UMISR register (C55FMC_UM9)	32	R/W	0000_0000h	34.4.17/ 1335
80	Over-Program Protection 0 register (C55FMC_OPP0)	32	R	See section	34.4.18/ 1336

Table continues on the next page...

C55FMC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
84	Over-Program Protection 1 register (C55FMC_OPP1)	32	R	See section	34.4.19/ 1337
88	Over-Program Protection 2 register (C55FMC_OPP2)	32	R	See section	34.4.20/ 1338
8C	Over-Program Protection 3 register (C55FMC_OPP3)	32	R	See section	34.4.21/ 1338
90	Test Mode Disable Password Check register (C55FMC_TMD)	32	R/W	0000_0000h	34.4.22/ 1339

34.4.1 Module Configuration Register (C55FMC_MCR)

NOTE

1. A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional Description.
2. See Functional Description for information about simultaneous MCR writes, and priority levels.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0											
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	FERS	0		PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

C55FMC_MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read.
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 A read-while-write error occurred during a previous read.
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections. 1 A Single Bit Correction occurred during a previous read.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that if the flash is sealed, and the Test Mode Disable Seal is programmed, erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations(and erase operations if the flash is unsealed). The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. **NOTE:** This bit transitions from a 0 to 1 during reset and remains at 1 after reset. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEG	Program/Erase Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. **NOTE:** 1. If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block. 2. If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set. 1 An interrupt request will be generated when the DONE flag is set.
24 FERS	Factory Erase and Program FERS is used for erase and program operations to enable a faster operation. Factory Erase must be done following “Initial Factory Conditions All Temps” with respect to voltage conditions, write/erase cycling, and temperature which can be from -40 °C to 150 °C;. After setting the ERS or PGM bit, but prior to setting EHV, FERS may be set to enable a Factory Erase or Program. FERS can be written only when ERS or PGM are high, but EHV is still low. Factory Erase may also be disabled by writing a location in the UTest NVM space. Address 0x0020 in the UTest NVM space will be checked at reset, and if programmed with data that contains at least one zero, a factory erase is not allowed, and FERS is locked. Writes attempted that violate the above cases will result in FERS remaining cleared. FERS is cleared on reset. 0 Factor Erase is disabled. 1 Factory Erase is enabled.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (ERS is low and UTE is low) • Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. The embedded flash memory enters suspend within Tpsus of this transition. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. The embedded flash memory enters suspend within Tesus of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: • Erase (ERS = 1, ESUS = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. **NOTE:** Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

34.4.2 Alternate Module Configuration Register (C55FMC_MCRA)

NOTE

1. This is an alternate interface register. It may be mapped to a different address than shown. See the Chip Configuration Chapter for details.
2. A number of Alternate Module Configuration Register (MCRA) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional Description .
3. See Functional Description for information about simultaneous MCRA register writes, and priority levels.
4. Reset values are dependent on chip configuration. See the Chip Configuration chapter for specifics.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				PEASa	DONEa	PEGa	0				PGMa	PSUSa	ERSa	ESUSa	EHVa
W																
Reset	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- Reset values are dependent on chip configuration. See the Chip Configuration chapter for specifics.

C55FMC_MCRA field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEASa	Program Access Space PEASa is used to indicate which space is valid for program operations, either main array space or UTest NVM space for the Alternate program and erase interface. It should be noted that erase is not allowed to the UTest NVM space (even if unsealed with the Test Mode Disable Seal) through the alternate interface. If an interlock write is done to the UTest NVM space with ERSa set, PEASa will remain 0. See PEAS definition in the MCR. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONEa	State Machine Status DONEa indicates if the embedded flash memory is performing a high-voltage operation for the Alternate program and erase interface. See DONE definition in the MCR for more information. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEGa	Program/Erase Good The PEGa bit indicates the completion status of the last flash memory program or erase sequence for which high-voltage operations were initiated for the Alternate program and erase interface. See PEG definition in the MCR for more information. It should be noted that in the event of a error on the alternate interface PEGa will clear, but the ADR will not be updated with the failing location. The ADR is only valid for main interface program or erase fails, and other fails noted in the ADR. 0 Program or erase operation failed. 1 Program or erase operation successful.
23–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGMa	Program PGMa is used to set up embedded flash memory for a program operation for the Alternate program and erase interface. See PGM definition in the MCR register for more information. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.
28 PSUSa	Program Suspend PSUSa is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state for the Alternate program and erase interface. See PSUS definition in the MCR register for more information. Erase ERSa is used to set up embedded flash memory for an erase operation for the Alternate program and erase interface. See ERS definition in the MCR register for more information. 0 Program sequence is not suspended. 1 Program sequence is suspended.
29 ERSa	Erase Suspend

Table continues on the next page...

C55FMC_MCRA field descriptions (continued)

Field	Description
	ERSa is used to set up embedded flash memory for an erase operation for the Alternate program and erase interface. See ERS definition in the MCR register for more information. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUSa	Erase Suspend ESUSa is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state for the Alternate program and erase interface. See ESUS definition in the MCR register for more information. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHVa	Enable High Voltage The EHVa bit enables the embedded flash memory for a high-voltage program/erase operation for the Alternate program and erase interface. See EHV definition in the MCR register for more information. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

34.4.3 Extended Module Configuration Register (C55FMC_MCRE)

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HT	0		n256K					n64Kh			n32Kh		n16Kh		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled. HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C and less and slower program and erase at 150 °C or less), and limited features at this temperature (no factory erase or or margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature. 1 165 °C maximum operating temperature with reduced performance.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 n256K	Number of 256 KB blocks. The value of this field is dependent upon the size of the embedded flash memory. This field is read only. 00000 Zero 256 KB blocks. 00001 Two 256 KB blocks (512 KB total). 00010 Four 256 KB blocks (1 MB total). 00011 Six 256 KB blocks (1.5 MB total). 00100 Eight 256 KB blocks (2 MB total). 00101 Ten 256 KB blocks (2.5 MB total). 00110 Twelve 256 KB blocks (3 MB total). 00111 Fourteen 256 KB blocks (3.5 MB total). 01000 Sixteen 256 KB blocks (4 MB total). 01001 Eighteen 256 KB blocks (4.5 MB total). 01010 Twenty 256 KB blocks (5 MB total). 01011 Twenty-two 256 KB blocks (5.5 MB total). 01100 Twenty-four 256 KB blocks (6 MB total). 01101 Twenty-six 256 KB blocks (6.5 MB total). 01110 Twenty-eight 256 KB blocks (7 MB total). 01111 Thirty 256 KB blocks (7.5 MB total). 10000 Thirty-two 256 KB blocks (8 MB total). 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Forty-eight 256 KB blocks (12 MB total). 11001 Reserved. 11010 Reserved. 11011 Reserved. 11100 Reserved. 11101 Reserved. 11110 Reserved. 11111 Reserved.
8–10 n64Kh	Number of 64 KB blocks in partitions 4 and 5. This field is read only.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Twelve 64 KB blocks.. 110 Sixteen 64 KB blocks. 111 Reserved.
11–12 n32Kh	Number of 32 KB blocks in partitions 4 and 5. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16–18 n64Km	Number of 64 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19–20 n32Km	Number of 32 KB blocks in partitions 2 and 3. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21–23 n16Km	Number of 16 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved.

Table continues on the next page...

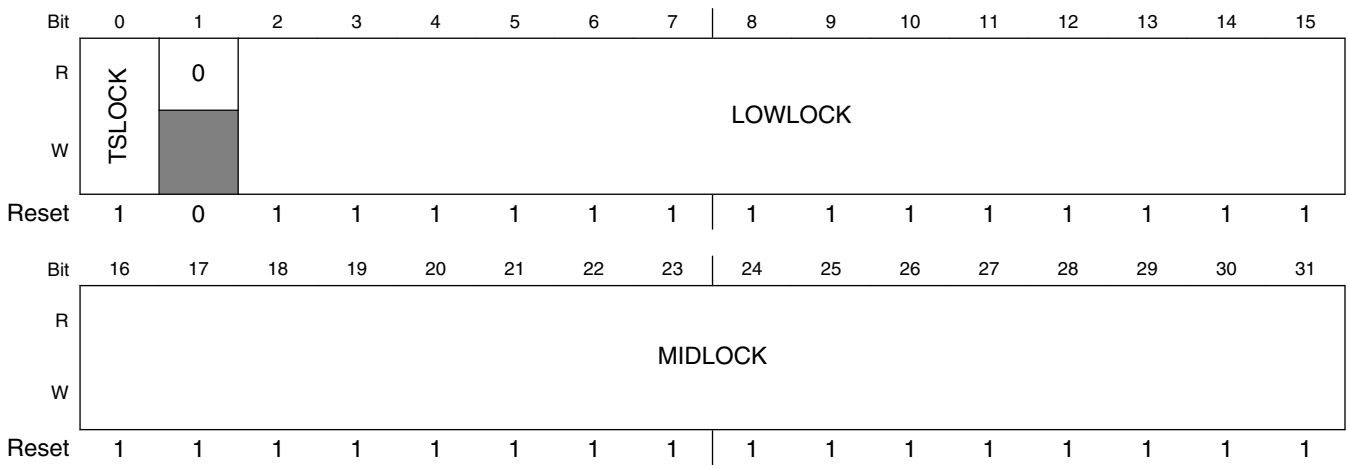
C55FMC_MCRE field descriptions (continued)

Field	Description
	110 Reserved. 111 Reserved.
24–26 n64KI	Number of 64 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
27–28 n32KI	Number of 32 KB blocks in partitions 0 and 1. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
29–31 n16KI	Number of 16 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.

34.4.4 Lock 0 register (C55FMC_LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

Address: 0h base + 10h offset = 10h



C55FMC_LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended. 0 UTest NVM block is available to be programmed. 1 UTest NVM block is locked and cannot be programmed.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended.

Table continues on the next page...

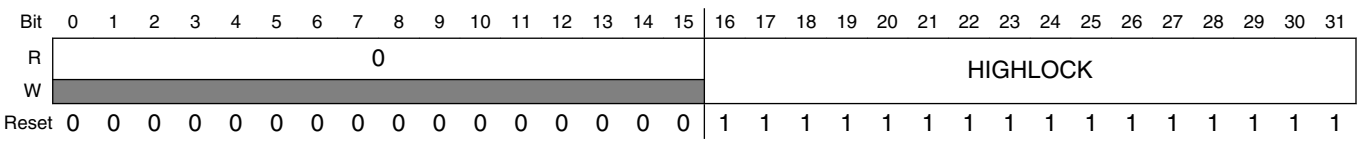
C55FMC_LOCK0 field descriptions (continued)

Field	Description
	The default value of the LOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

34.4.5 Lock 1 register (C55FMC_LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Address: 0h base + 14h offset = 14h



C55FMC_LOCK1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

34.4.6 Lock 2 register (C55FMC_LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KLOCK																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK2 field descriptions

Field	Description
0–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

34.4.7 Lock 3 register (C55FMC_LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

34.4.8 Alternate Lock 0 register (C55FMC_LOCK0A)

The Alternate Lock 0 (LOCK0a) register provides a means to protect blocks from being modified on the alternate program and erase interface.

NOTE

This is an alternate interface register. It may be mapped to a different address than shown. See the Chip Configuration information for details.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCKa	0	LOWLOCKa													
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCKa															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK0A field descriptions

Field	Description
0 TSLOCKa	Alternate UTest NVM Lock. This bit is used to lock the UTest NVM block from programs on the alternate program and erase interface (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCKa register signifies that the UTest NVM block is locked for program on the alternate interface. A value of 0 in the TSLOCKa register signifies that the UTest NVM block is available to receive program pulses on the alternate interface. The TSLOCKa register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, TSLOCKa register is not writable if a high voltage operation is suspended. The TSLOCKa bit may be written as a register. The default value of the TSLOCKa bits is locked. 0 The UTest NVM block is available to receive program pulses on the alternate interface. 1 The UTest NVM block is locked for program on the alternate interface.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCKa	Alternate Low Block Lock This is the method to lock blocks for the alternate program and erase interface. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCKa[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCKa bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCKa bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCKa	Alternate Mid Block Lock This is the method to lock blocks for the alternate program and erase interface. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCKa[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3.

Table continues on the next page...

C55FMC_LOCK0A field descriptions (continued)

Field	Description
	The lock register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCKa bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCKa bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

34.4.9 Alternate Lock 1 register (C55FMC_LOCK1A)

The Alternate Lock 1 (LOCK1a) register provides a means to protect blocks from being modified on the alternate program and erase interface.

NOTE

This is an alternate interface register. It may be mapped to a different address than shown. See the Chip Configuration information for details.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCKa															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK1A field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCKa	Alternate High Block Lock This is the method to lock blocks for the alternate program and erase interface. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCKa[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted.

Table continues on the next page...

C55FMC_LOCK1A field descriptions (continued)

Field	Description
	HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONEa is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCKa bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writeable. This enables blocks to be present in only one program and erase space (main or secondary). In the event that blocks are not present (due to configuration or total memory size), the LOCKa bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

34.4.10 Select 0 register (C55FMC_SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWSEL														MIDSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWSEL	LOW Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1.

Table continues on the next page...

C55FMC_SEL0 field descriptions (continued)

Field	Description
	The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase
16–31 MIDSEL	Mid Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

34.4.11 Select 1 register (C55FMC_SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

C55FMC_SEL1 field descriptions (continued)

Field	Description
16–31 HIGHSEL	High Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected for erase 1 The block is selected for erase

34.4.12 Select 2 register (C55FMC_SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL2 field descriptions

Field	Description
0–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect.

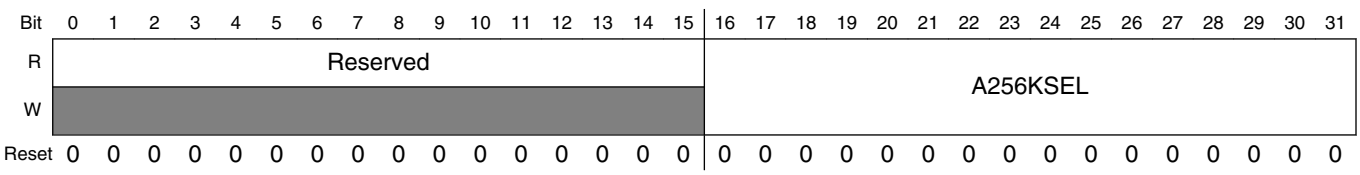
C55FMC_SEL2 field descriptions (continued)

Field	Description
	0 The block is not selected 1 The block is selected for erase

34.4.13 Select 3 register (C55FMC_SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 44h offset = 44h



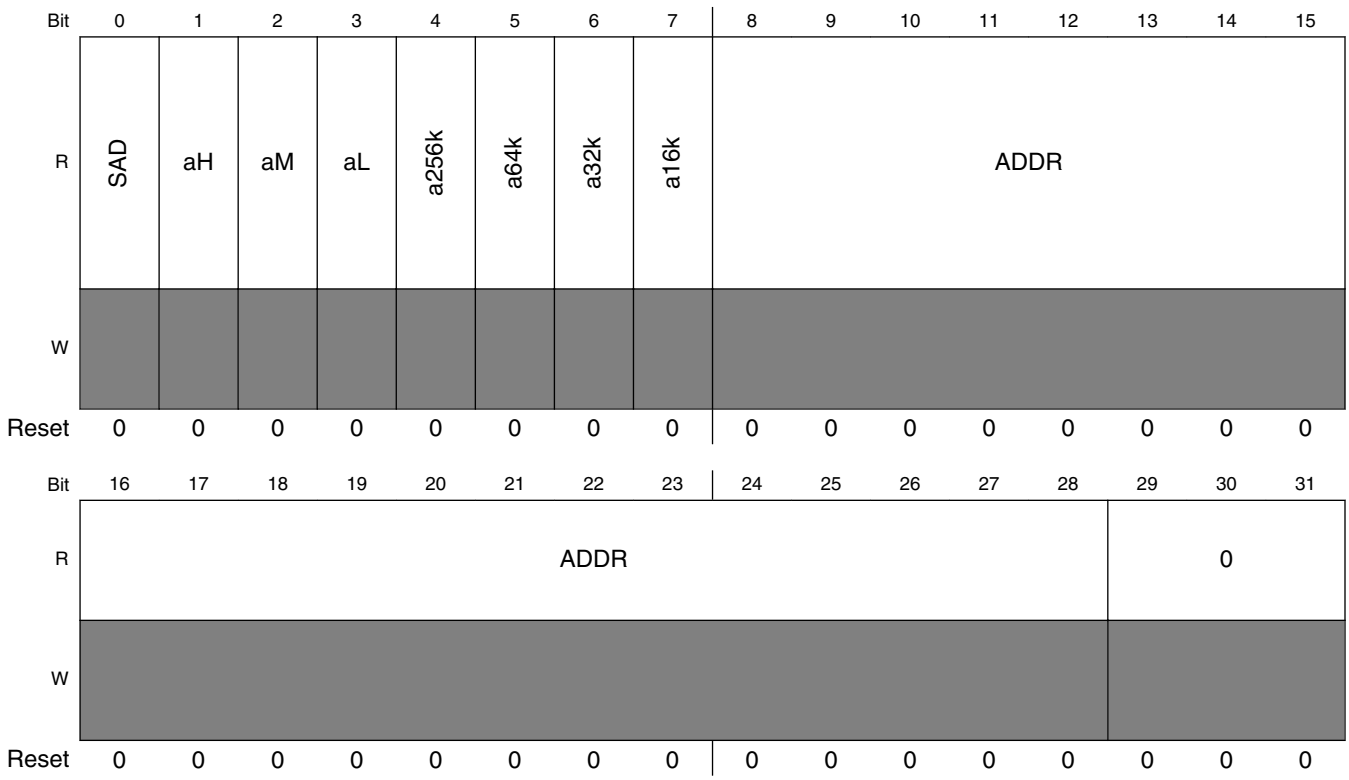
C55FMC_SEL3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved, reset to 0.
16–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 1 The block is selected for erase. 0 The block is not selected.

34.4.14 Address register (C55FMC_ADR)

The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

Address: 0h base + 50h offset = 50h



C55FMC_ADR field descriptions

Field	Description
0 SAD	UTest NVM Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD field is not writable. 0 Address captured is from main array Space. 1 Address captured is from UTest NVM array space.
1 aH	Address High region. The aH bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region. 1 Address captured or to be accessed is from high address region.
2 aM	Address Mid region. The aM bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information.

Table continues on the next page...

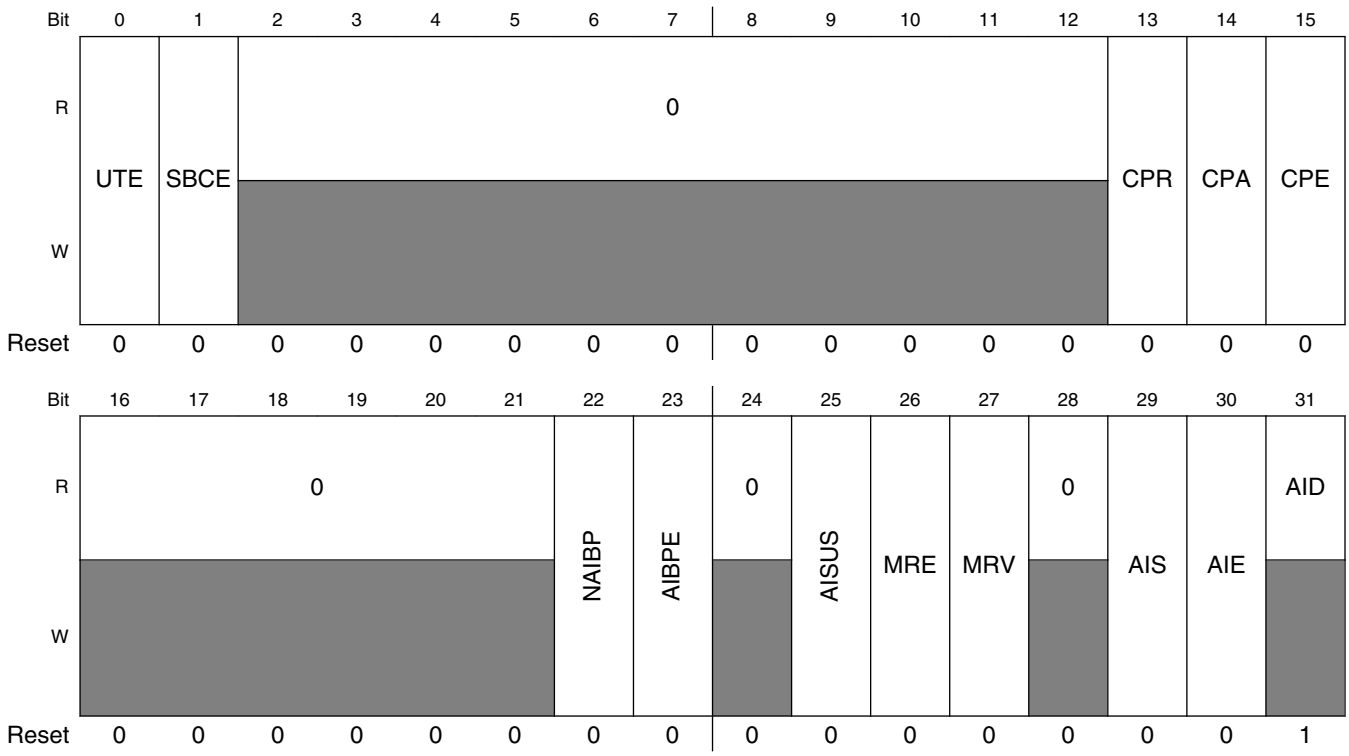
C55FMC_ADR field descriptions (continued)

Field	Description
	0 Address captured or to be accessed is not from mid address region. 1 Address captured or to be accessed is from mid address region.
3 aL	Address Low region. The aL bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region. 1 Address captured or to be accessed is from low address region.
4 a256k	Address 256 KB region. The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region. 1 Address captured or to be accessed is from 256 KB region.
5 a64k	Address 64 KB region. The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region. 1 Address captured or to be accessed is from 64 KB region.
6 a32k	Address 32 KB region. The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region. 1 Address captured or to be accessed is from 32 KB region.
7 a16k	Address 16 KB region. The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region. 1 Address captured or to be accessed is from 16 KB region.
8–28 ADDR	Address. The ADR provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.4.15 UTest 0 register (C55FMC_UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

Address: 0h base + 54h offset = 54h



C55FMC_UT0 field descriptions

Field	Description
0 UTE	U-Test Enable. This status bit gives indication when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. 0 All bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are locked 1 Bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are unlocked
1 SBCE	Single Bit Correction Enable. SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	0 Single Bit Corrections observation is disabled. 1 Single Bit Correction observation is enabled.
2–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CPR	Customer Programmable Read Voltage and Reference Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made o write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 NAIBP	Next Array Integrity Break Point. If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 1 Array Integrity state machine is at a break point. 0 Array Integrity state machine is not currently at a break point.
23 AIBPE	Array Integrity Break Point Enable. If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 1 Array Integrity breakpoints are enabled during Array Integrity Checks. 0 Array Integrity breakpoints are not enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AISUS	Array Integrity Suspend. AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by Setting AISUS, and resumed by clearing AISUS. AISUS is writeable only when AID is low. AISUS is clearable only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	1 Array integrity sequence is suspended. 0 Array integrity sequence not suspended.
26 MRE	Margin Read Enable. MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 Margin reads are enabled. 0 Margin reads are not enabled.
27 MRV	Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 One's margin reads are requested. 0 Zero's margin reads are requested.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AIS	Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. 1 Array integrity sequence is sequential. 0 Array integrity sequence is proprietary sequence.
30 AIE	Array Integrity Enable. AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled. 1 Array integrity checks are enabled.
31 AID	Array Integrity Done. AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing. 1 Array integrity check is done.

34.4.16 UMISR register (C55FMC_UM*n*)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 58h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM*n* field descriptions

Field	Description
0–31 MISR	MISR[31:0]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

34.4.17 UMISR register (C55FMC_UM9)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															MISR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM9 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MISR	MISR[288]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained

34.4.18 Over-Program Protection 0 register (C55FMC_OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWOPP														MIDOPP															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWOPP	Low Block Over-Program Protection[13:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.
16–31 MIDOPP	Mid Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The MIDOPP register is not writable, and is status only.

Table continues on the next page...

C55FMC_OPP0 field descriptions (continued)

Field	Description
	The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

34.4.19 Over-Program Protection 1 register (C55FMC_OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHOPP															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHOPP	High Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

34.4.20 Over-Program Protection 2 register (C55FMC_OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP2 field descriptions

Field	Description
0–31 A256KOPP	256K Block Over-Program Protection[31:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 0 The corresponding block is available to be over-programmed.

34.4.21 Over-Program Protection 3 register (C55FMC_OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KOPP															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KOPP	256K Block Over-Program Protection[47:32]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

34.4.22 Test Mode Disable Password Check register (C55FMC_TMD)

The Test Mode Disable Password Check (TMD) register provides a means to provide a challenge password to disable the Test Mode Disable Seal block select.

Writes to the register have no effect except for a password challenge. Reads to this register always return 0.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PWD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_TMD field descriptions

Field	Description
0–31 PWD	Password challenge

34.5 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in Module Configuration Register (C55FMC_MCR). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 34-2. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- Low Power mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

34.5.1 User Mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

34.5.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read, except when the embedded flash memory is in low power mode. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

NOTE

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

NOTE

Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] =0) will result in the interlock write being ignored, and address and data will not be updated.

NOTE

Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See the Module Configuration Register (MCR) section for more information.

34.5.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning

In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

NOTE

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated as an Over-Program Protected block.

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

   NOTE

   Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as

a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the Flash Memory Controller chapter for information on the size of writes that are allowed.

3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning

Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

34.5.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

34.5.1.2.2 Program hardware locking

Hardware locking which only affects the program operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

34.5.1.2.3 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in the Module Configuration Register (MCR) section. Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

NOTE

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0). Although suspend frequency in not limited, enabling at least 20uS between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation.

34.5.1.2.4 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

34.5.1.2.5 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset. This feature is valid for programs done on the main interface, and not valid for programs done on the alternate interface.

34.5.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.
2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed if sealed (since erase is not allowed in the UTest NVM space when sealed).
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored(except for alternate erase interface). The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

34.5.1.3.1 Erase software locking

Software Locking affect erase operation. For details on this see Program software locking.

34.5.1.3.2 Erase hardware locking

Hardware locking which only affects the erase operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

34.5.1.3.3 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, provided the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program) before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Warning

In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.

Warning

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0). Although suspend frequency is not limited, enabling at least 5mS between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation.

34.5.1.4 Alternate program and erase interface

A second interface is provided on the flash for program and erase operations. The intent is that a dedicated master can have private access to blocks through this interface.

The alternate interface includes an alternate MCR register and alternate lock registers. Any block in low, mid, or high address space may be accessible through the alternate interface. The UTest NVM Block is also accessible for program through the alternate interface. 256 KB sized blocks are not accessible through this interface.

The alternate interface supports programming of data sizes from one doubleword (64 bits) to four doublewords within a single page (256 bits). Within a page, up to eight words may be altered in a single program operation. It is recommended that programming operations through the alternate interface be from 64 bits to 256 bits, and be 64-bit aligned.

The alternate interface supports erasing a single block at a time with each erase operation.

As with the main interface, erase suspend, program suspend, and erase-suspended programs are supported.

Program and Erase times for the alternate interface are the same as the main interface when programs and erases are done in isolation. If main interface and alternate interface are being utilized simultaneously, bandwidth sharing is utilized.

NOTE

It is possible for the UTest NVM block to exist in both the main program interface and alternate program interface. In the event of a conflict the first interface to do the program to an address will get priority, and all other attempts after this will result in an OTP over-program failure (MCR[PEG] or MCRA[PEGa] being set to 0).

34.5.1.4.1 Alternate program interface

The programming procedure follows exactly the same flow as the main interface, except that only page programming is allowed. See section Program for more information on programming.

The alternate interface program is done at lower priority compared to the main interface. Based on the size of programming allowed, the main interface has a 4:1 ratio advantage over the alternate interface. Program operations are allowed to finish on the active interface prior to allowing the inactive interface to execute its operation. Suspends done on the active interface enable the inactive interface to begin its operation.

NOTE

For alternate program while erase, there could be a latency of up to 5 ms in the program operation.

For more information on interaction between the alternate program interface and the main interface, see Alternate interface performance.

If the alternate interface is the only interface active, then normal program times may be expected.

34.5.1.4.2 Alternate erase interface

For erases done thru the alternate interface, the single block to be erased is selected based on the address that is received during the erase interlock write. Thus the Select registers are not valid for the alternate interface (step 2 of the erase flow in section Erase), and the interlock write address is used (step 3 of the erase flow in section Erase). All other features of the main interface apply to the alternate erase interface. The erase procedure is exactly the same as for the main interface. See section Erase for more information on erase.

The alternate interface erase is done at lower priority compared to the main interface. The time-slice ratio between the two interfaces is 10:1, with 50 ms of time given to the main erase for every 5 ms of alternate interface. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate erase while program, there can be a latency of up to 5 ms in the main program time. Once the alternate erase is interrupted, it will remain interrupted for main programs for a fixed time of 50 ms, at which time the alternate erase will restart. It should be noted that multiple interruptions of the alternate erase will significantly increase the total erase time.

For alternate erase while erase, the main erase has priority and receives 50 ms of erase for every 5 ms of alternate erase. This may result in a significant increase in the erase time for the alternate interface, and a minor increase in erase time for the main interface.

For more information on interaction between the alternate erase interface and the main interface, see Alternate interface performance

If the alternate interface is the only interface active, then normal erase times may be expected.

34.5.1.4.3 Alternate interface performance

Table 34-3 highlights the interactions between the alternate interface and the main interface for program, erase, suspend, and idle conditions.

Table 34-3. Alternate program and erase characteristics

Alt PGM/ERS interface request	Main PGM/ERS interface request	Alt interface result on memory array	Main interface result on memory array
IDLE	IDLE	IDLE	IDLE
IDLE/SUS	PGM/ERS	IDLE/SUS	PGM/ERS
PGM/ERS	IDLE/SUS	PGM/ERS	IDLE/SUS
PGM	PGM	PGM	Priority PGM (up to 4:1)
PGM	ERS	PGM with latency of 5 ms	ERS (possible minimal increase to total ERS time)
ERS	PGM	ERS (significant increase in total ERS time expected)	Priority PGM (possible latency of 5 ms) Once PGM is granted, 50 ms time slice available for more programs
ERS	ERS	ERS (significant increase in total ERS time expected)	Priority ERS (10:1) (possible minor increase to total ERS time) 50 ms Main ERS for every 5 mS alternate ERS

34.5.2 Low Power mode

In Low Power mode, the embedded flash memory enables VDDF power gating, and disables circuits on the VFLASH domain. No reads from or writes to the embedded flash memory are possible when in Low Power mode.

Prior to entering Low Power Mode, it is required that all program, erase, and UTest operations be stopped prior to entering the mode. If Low Power mode is entered while an operation is suspended, entering Low Power mode will have the same effect as a reset during suspend.

When in Low Power mode, register access is prevented. Flash core accesses are also prevented until power mode is exited. Flash core reads and writes may occur after power mode is exited.

The embedded flash memory returns to a post-reset state in all cases.

34.5.3 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

34.5.3.1 Array integrity self check

Array integrity is checked using a predefined address sequence (based on UT0[AIS]), and this operation is executed on selected blocks. The data to be read is customer-specific - user code may be programmed into the flash memory and the correct MISR signature is calculated based on that code. Any random or nonrandom code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value, to determine if an incorrect read or ECC detection was noted. Array integrity MISR value is calculated after ECC detection and correction. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the system frequency being used. The array integrity check consists of the following sequence of events:

1. Enable UTest mode.
2. Select the block or blocks to be receive the array integrity check by writing 1's to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for array integrity check do not need to be unlocked.

NOTE

Unselected blocks are still read as part of the array integrity sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

NOTE

Blocks protected with the Test Mode Disable Seal are still read as part of the array integrity sequence. The resulting read on sealed blocks will not be captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints will still be honored.

NOTE

It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

NOTE

For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that AIS be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, ensure that UT0[SBCE] is set.
6. Set the UT0[AIE] bit.
   a. If desired, the array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and then continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers will contain a signature for the portion of the operation that was completed prior to the abort, and will not be deterministic. Prior to doing another array integrity operation, the UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9 registers may need to be initialized to the desired seed value by doing register writes.
   b. If desired, the array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After

the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which indicates the flash has entered the suspend state, and normal reads to the flash may be done. Once [AID] goes high, UT0[AISUS] may be cleared to resume the array integrity sequence.

NOTE

User mode array reads requested during the array integrity test will be ignored, to ensure that the array integrity operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

7. Wait until the UT0[AID] bit goes high.
8. If breakpoints were enabled, check UT0[NAIBP], and if this is set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear MCR[SBC] or MCR[EER] bits. Then the operation may be resumed by clearing UT0[NAIBP]. Continue to wait until the UT0[AID] bit goes high. If breakpoints were not enabled, or if UT0[NAIBP] is low when UT0[AID] goes high, then the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit.

34.5.3.2 User margin read

User margin read may be done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin reads are done at a read margin level checking for erased bits or programmed bits encroaching on the nominal read level. User margin read is a self-timed event, and is independent of system clocks or wait states selected. Margin ECC corrections and detections are noted during the user margin read test. Margin read MISR value is calculated after ECC detection and correction.

The data to be read is customer-specific, and user code may be programmed into the flash memory. Any random or non-random code is valid. Once the operation is completed, the margin read results can be checked by reading the MCR[EER] bit and the MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, as well as checking the MISR.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode.
2. Select the block or blocks to receive margin read check by writing 1's to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for margin read do not need to be unlocked.

NOTE

Unselected blocks are still read as part of the margin read sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks will not result in a faster margin read execution time.

NOTE

Blocks protected with the Test Mode Disable Seal are still read as part of the margin read sequence. The resulting read on sealed blocks will not be captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints will still be honored.

NOTE

It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.

NOTE

For margin read checks of the flash memory, sequential addressing is recommended. Setting AIS to 0 is possible for Margin Reads, but using the sequence takes more time, and is not recommended.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. Ensure that the MCR[EER] and MCR[SBC] bits are cleared.
6. If you want to detect single bits during margin read, set UT0[SBCE] to 1.

7. Set the UTO[MRE] bit.
8. Set the UT0[MRV] bit to the desired value depending on if it is desired to do one's margin or zero's margin.
9. Set the UT0[AIE] bit.

NOTE

User mode array reads requested during the margin read test will be ignored, to ensure that the margin read operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

NOTE

During Margin Read operations, with AID low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

10. Wait until the UT0[AID] bit goes high.
    a. If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Array integrity self check for more information.
11. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

12. Write a logic 0 to the UT0[AIE] bit.

34.6 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash

memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning

Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

Chapter 35
Decorated Storage Memory Controller (DSMC)

35.1 Introduction

This section details the hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called "*decorated storage*". It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A "decoration" is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rB,rA` where the data size specifier is defined as 8-bit (`b` = byte), 16-bit (`h` = halfword) or 32-bit (`w` = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rB,rA` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `"x"` mnemonic suffix, signaling an "indexed" addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in the following figure.

Table 35-1. Power Architecture decorated load and store instruction formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	rT						rA				rB				1	0	0		1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0		1	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		1	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit "decoration". The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and/or operand data for the command.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

35.2 Decorated stores: `st[b,h,w]d{cb}x rS,rA,rB`

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a `<command>.<size>` syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA

- accessAddress = rB
- wdata = rS

35.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit memory container

Table 35-2. Decoration format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory "container" defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read
if bfw == 0
         then bfw = container

if ((srtbit + bfw) <= container) // generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp  & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp // memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
```

```
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

35.2.1.1 Compare-and-Store (CAST)

Table 35-3. Decoration format: CAST

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CD8							
cast.h	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	CD16															
cast.w	1	0	0	1	CD28																											

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size]  // memory read

if (size == 8)             // define compare_operand
    compare_operand = CD8
else if (size == 16)
         compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand) // compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

35.2.1.1.1 Logical AND (AND)

Table 35-4. Decoration format: AND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Table 35-4. Decoration format: AND (continued)

and.{b,h,w}	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata      // modify
mem[accessAddress, size] = tmp// memory write
```

35.2.1.1.2 Logical OR (OR)

Table 35-5. Decoration format: OR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
or.{b,h,w}	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata          // modify
mem[accessAddress, size] = tmp// memory write
```

35.2.1.1.3 Logical Exclusive-OR (XOR)

Table 35-6. Decoration format: XOR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
xor.{b,h,w}	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata          // modify
mem[accessAddress, size] = tmp// memory write
```

35.2.1.2 Decorated Loads: `l[b,h,w]d{cb}x rT,rA,rB`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an "extract") operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

35.2.1.2.1 Simple Load (SLD)

Table 35-7. Decoration format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld.{b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

35.2.1.2.2 Registers-and-memory exchange (SWAP)

Table 35-8. Decoration format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
swap.b	0	1	0	1	-	-	**-**	-	-	-	-	**-**	-	-	-	-	**-**	-	-	-	-	-	-	-	WD8							
swap.h	0	1	0	1	-	-	-	-	-	-	-	**-**	-	-	-	-	**WD16**															
swap.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)             // store_operand - "modify"
    store_operand = WD8
else if (size == 16)
           store_operand = WD16
    else store_operand = {0x0, WD28} // zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit store operand.

35.2.1.2.3 Load-and-Set-1(Bit) (LAS1)

Table 35-9. Decoration format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp // rT = rdata = tmp
mask  = 1 << (container - bit - 1) // generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

35.3 DSMC instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

Chapter 36
Flash Memory Programming and Configuration

36.1 Introduction

This chapter provides basic guidance on activities related to configuring and programming the embedded flash memory in the MPC5777M microcontroller.

Multiple layers of protection are available to prevent unintended updates to flash contents and unauthorized reads and writes to flash contents. The protection mechanisms range from simple write-protect bits for individual flash blocks to four levels of 256-bit password read and/or write protection and configuration of flash memory blocks as One Time Programmable (OTP).

Additionally, the MPC5777M microcontroller includes a tamper detection feature, debug port enable/disable, and the ability to disable the ability for the device to be put into factory test mode.

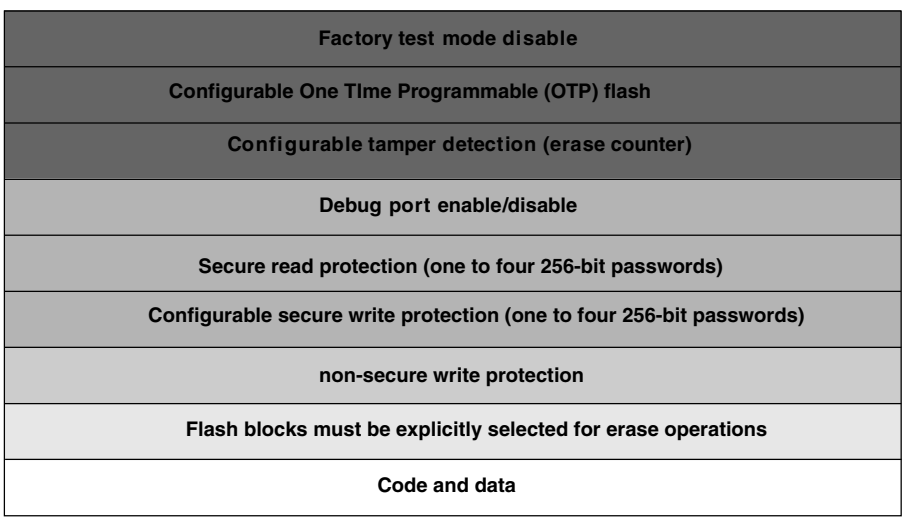


Figure 36-1. Flash memory protection layers

At the lowest layer of protection, all flash erase operations require the explicit selection of one or more blocks of flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured *write* protection for individual flash blocks.

Similarly, up to four levels of password-secured *read* protection can be implemented. Unlike the password-secured write protection, which is implemented on a per-block basis, password-secured read protection applies to five pre-defined (at the factory) groups of flash blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

The MPC5777M microcontroller also includes a tamper detection feature. In its most basic implementation, tamper detection acts as an erase counter. If tamper detection is implemented, a DCF record in the tamper detection *diary* must be written by software before a protected flash block can be erased. Multiple tamper detection regions can be defined by the customer, with each region containing one or more blocks. A block can be included in more than one tamper detection region if desired.

Another protection feature related to tamper detection is the ability to configure any flash block as one-time programmable (OTP).

Finally, the microcontroller includes a feature that allows the customer to disable the ability to place the device in factory test mode.

The remainder of the chapter provides details regarding these protection mechanisms and gives a brief overview of security configuration planning.

Note

Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the flash protection features work and relate to each other, please read the chapter in sequence from beginning to end.

Access protection configuration is applied on a per-block basis (except read protection and tamper detection applied to groups of flash memory blocks), i.e., a custom level of access protection can be provided for each flash memory block. This is accomplished

through the use of registers containing bits that are mapped to individual flash blocks. Understanding the mapping is key to understanding the implementation and use of the MPC5777M microcontroller's flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in Selection of flash memory blocks for erase. It is important to understand the topic before proceeding to other sections.

36.2 Selection of flash memory blocks for erase

The registers used for selecting flash blocks for erase operations are the flash control registers named LOCK0, LOCK1, and LOCK2. There is also a LOCK3 register, but no flash blocks are mapped to it.

The LOCK0–2 registers are 32-bit registers containing fields associated with specific address spaces: low, mid, high, and 256 KB. Each non-reserved bit in the fields is mapped to a specific flash block.

For example, Figure 36-2 shows the Select 0 (SEL0) register for the flash module. This register provides a way of selecting blocks in the low and mid address spaces for erase operations.

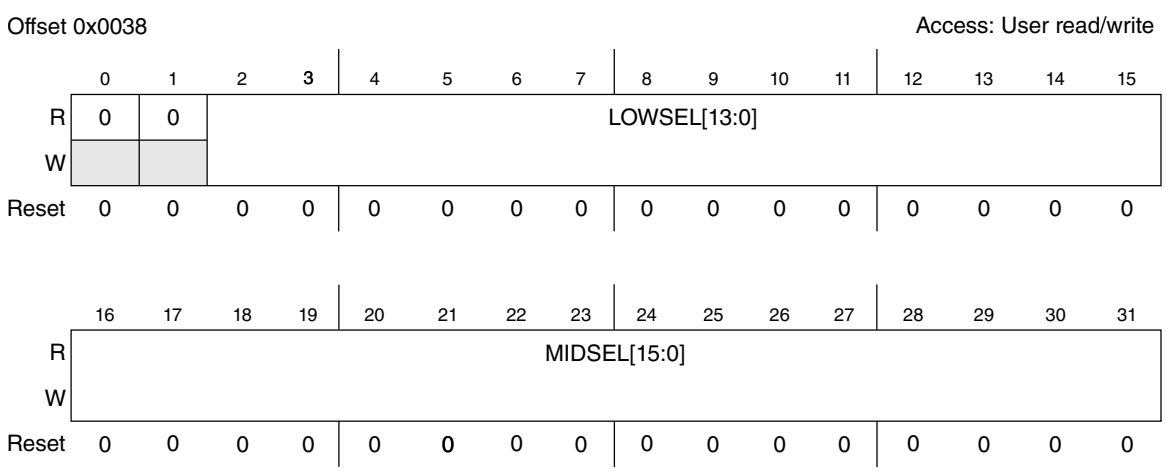


Figure 36-2. SEL0 flash control register

Table 36-1 shows an example of a mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual flash blocks[1]. In this case, not all bits of the LOWSEL and MIDSEL fields are mapped—only LOWSEL bits 0–11 (register bits 15–4,[2]) and

1. The table lists the LOWLOCK register field but the structure is identical to the LOWSEL field in Figure 36-2.
2. Register bits in Power Architecture-based microcontrollers are numbered with the most significant bit as 0. This is not necessarily the case with register field bit numbering. The field bit numbering shown in the figure has the least significant bit numbered as 0.

MIDSEL bits 0–2 are mapped. Before any mapped block can be erased, it must be selected by writing a '1' to its corresponding bit in one of the SEL*n* registers. See the device configuration chapter for the complete mapping.

Note

Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in Figure 36-2, all bits are reset to '0'. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Table 36-1. Sample low and mid address space flash block bit mapping

Start Address	End Address	Size (KB)	Partition	LOCK*n*[1] Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit[2]	Read Lock group[3]	Block No.	Description
0x0040_4000	0x0040_7FFF	16	0	LOCK0	LOWLOCK[0]	16	—	0	BAF[4]
0x00FC_0000	0x00FC_3FFF	16	0	LOCK0	LOWLOCK[1]	17	1	1	Code Flash[5]
0x00FC_4000	0x00FC_7FFF	16	0	LOCK0	LOWLOCK[2]	18	1	2	Code Flash[4]
0x00FC_8000	0x00FC_BFFF	16	1	LOCK0	LOWLOCK[3]	19	1	3	Code Flash[4]
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash[4]
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code
0x00FD_0000	0x00FD_7FFF	32	0	LOCK0	LOWLOCK[6]	22	1	6	Code Flash
0x00FD_8000	0x00FD_FFFF	32	1	LOCK0	LOWLOCK[7]	23	1	7	Code Flash
0x00FE_0000	0x00FE_FFFF	64	0	LOCK0	LOWLOCK[8]	24	1	8	Code Flash
0x00FF_0000	0x00FF_FFFF	64	0	LOCK0	LOWLOCK[9]	25	1	9	Code Flash
0x0061_0000	0x0061_FFFF	64	1	LOCK0	LOWLOCK[10]	26	3	10	Secure Code
0x0062_0000	0x0062_FFFF	64	1	LOCK0	LOWLOCK[11]	27	3	11	Secure Code
0x0068_0000	0x0068_3FFF	16	2	LOCK0	MIDLOCK[0]	0	4	12	Secure Data
0x0068_4000	0x0068_7FFF	16	3	LOCK0	MIDLOCK[1]	1	4	13	Secure Data
0x0040_0000	0x0040_3FFF	16	0	LOCK0	TSLOCK	31	0	32	UTEST[6]

1. There are multiple sets of LOCK*n* registers, i.e., the flash module's LOCK0–-3 registers and PASS module's PASS_LOCK0–3_PG0–3, but all use the same mapping of register bits to flash blocks. The same mapping applies the the flash module's SEL0–3 registers.
2. The DCF record DATA bit refers to the DATA[31:0] field contained in DCF records created during OEM configuration activities. The DCF records related to flash blocks are used to implement password-secured write protection, password-secured read protection, tamper detection, and debug port enable/disable as a permanent part of the device configuration.
3. The Read Lock group pertains to protecting code and data blocks from being read by debuggers. This type of protection is implemented using DCF records and the PASS_LOCK3_PG0–3 registers.
4. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.
5. This block is searched to determine the boot location (if it contains the boot header).
6. One time programmable.

Example: 36.2.1 Selecting blocks for erase

Selecting blocks for erase

Referring to Table 36-1, writing the value 0x0030_0000 to the SEL0 register (Figure 36-2) selects the following blocks for erase:

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash[4]
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

Note

The mappings of SEL*n* register bits to flash blocks vary with different microcontrollers. See the flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the MPC5777M microcontroller.

36.3 Non-secure write protection

Non-secure write protection refers to the chip's ability to lock individual flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is *not* associated with the built-in register protection (REG_PROT) feature included with many of the chip's modules. The non-secure write protection feature detailed here applies only to embedded flash memory.

Implementing non-secure write protection is almost identical to selecting specific flash blocks for an erase operation—the only difference is the register used. To provide non-secure write protection for a block, write a '1' to the corresponding mapped bit found in one of the LOCK0–2 flash control registers.

Note

As indicated by the reset values shown in Figure 36-3, after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The same bit mapping that applies to the SEL0–3 registers also applies to the LOCK0–3 registers.

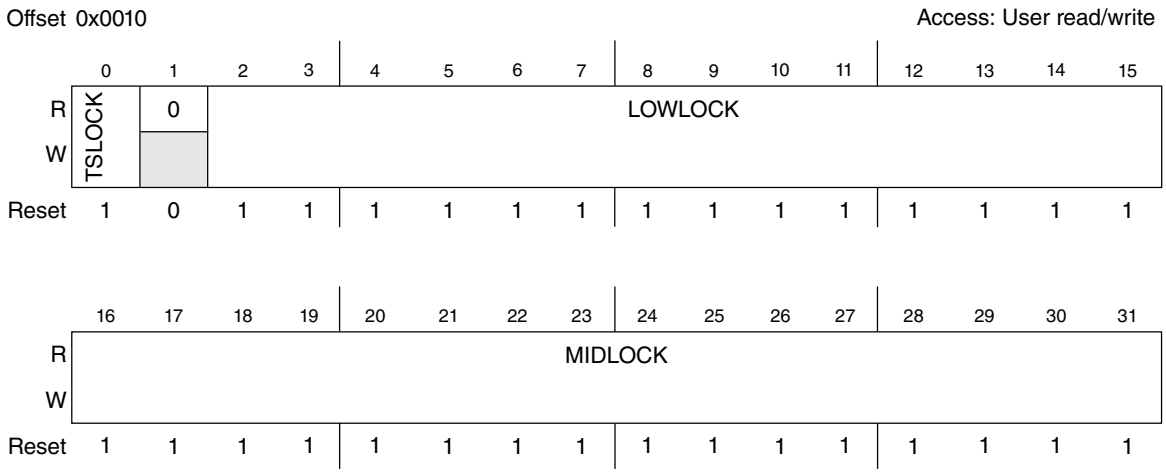


Figure 36-3. LOCK0 flash control register

Example: 36.3.1 Selecting

Selecting

Referring to Table 36-2, writing the value 0x0030_0000 (same value used in Selecting blocks for erase) to the LOCK0 register (Figure 36-3) locks the following blocks (same blocks selected in Selecting blocks for erase) so they cannot be programmed or erased, even if selected for erase:

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash[4]
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in Secure write protection). All other blocks mapped to bits in the LOCK0 *can* be programmed because their corresponding bits have been set to '0'.

36.4 Secure write protection

Secure write protection refers to the chip's ability to implement up to four levels of 256-bit password-secured *write* protection for individual flash blocks. Use of this feature is optional but the default behavior of the MPC5777M microcontroller is to apply secure write protection. If there is a need to *not* write-protect specific blocks, that needs to be configured by creating the appropriate DCF records.

Note that the *secure* write protection settings override the non-secure write protection detailed in Non-secure write protection.

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each flash block. The DCF records use a bit mapping scheme identical to the mapping in the flash module's SEL*n* and LOCK*n* registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers are four identical sets of registers contained in the PASS module—they distinguished by the PG*n* suffix. The PG*n* (Password Group *n*) suffix indicates the password required to unlock that set of registers. Password 0 unlocks PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. Similarly, the registers suffixed with PG1 are unlocked by matching password 1, and so on.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding flash block.

Each set of registers contains the same mapping of bits to flash blocks as the flash module's SEL*n* and LOCK*n* registers. In any of the four PASS_LOCK*x*_PG*n* register sets, if a mapped bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one

PASS_LOCK*x*_PG*n* register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

Figure 36-4 shows how the registers interact. Note that secure write protection is only affected by the settings in PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n*. The PASS_LOCK3_PG*n* registers have additional security functions, discussed in later topics.

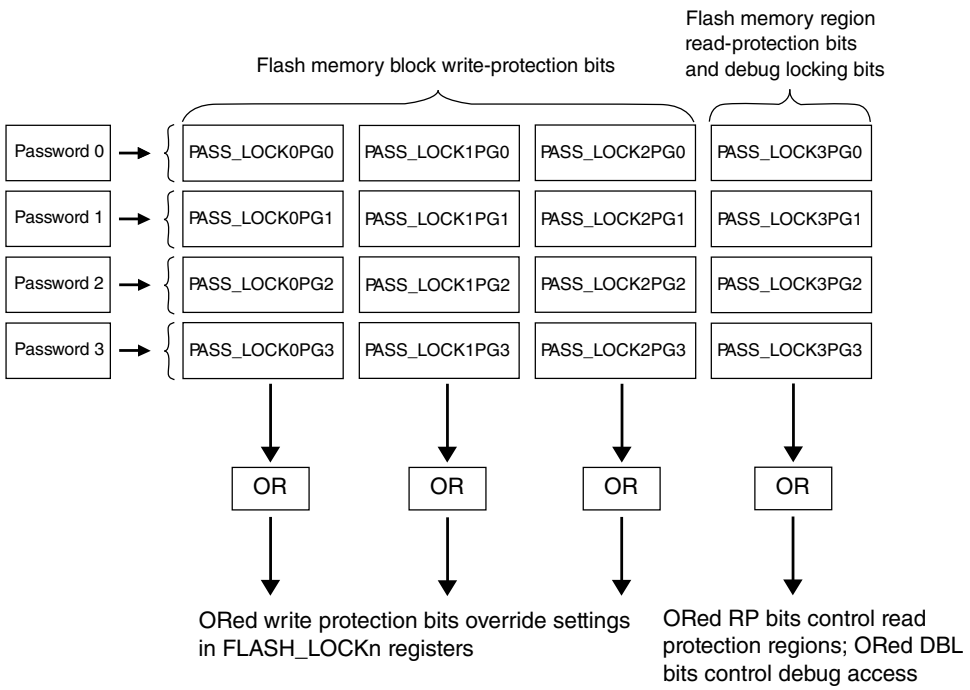


Figure 36-4. Secure write protection

The next sections discuss implementing secure write protection and overriding secure write protection.

36.4.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers[3]. The DCF records have a structure that includes the same mapping of bits to flash memory blocks as

3. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

the flash module's SEL*n* and LOCK*n* registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers.

See Device Configuration Format (DCF) Records for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to flash memory blocks is identical to the mapping used for selecting flash blocks for erase of non-secure write locking, there are two significant differences:

1. Configuring flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

CAUTION

Careful planning is required before writing DCF records. They are created in a write-once area of flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of DCF client list table (Table 9-6 in Device Configuration Format (DCF) Records). Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST flash memory. The structure of a DCF Record is shown in Figure 36-5.

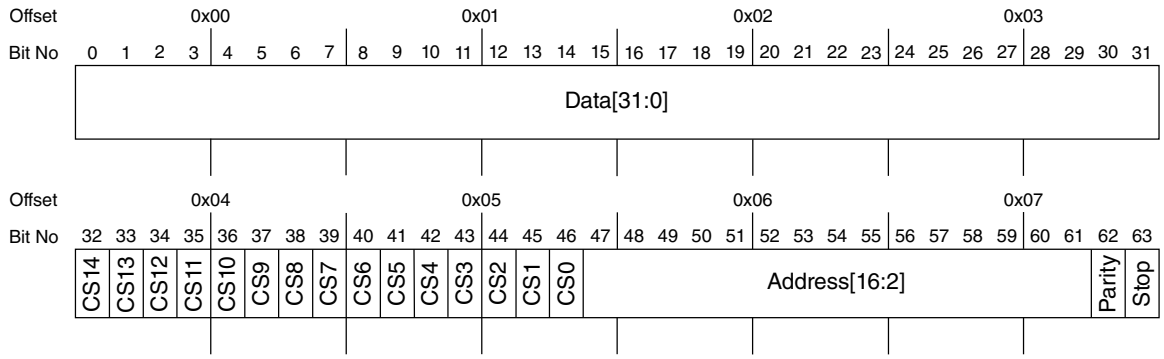


Figure 36-5. DCF record structure

Note

1. The values in the PASS_LOCK*x*PG*n* registers override the settings in the flash module's LOCK0–3 registers.
2. Values written to the PASS_LOCK*x*PG*n* registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase.
3. The PASS_LOCK*x*PG*n* bits take effect when the life cycle is matured to "OEM Production" or older.

Example: 36.4.1.1 Implementing secure write protection

Implementing secure write protection

Implementation of secure write protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) must be stored in a protected OTP area of device flash called UTEST flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n* –PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers.
2. DCF records must be created to define which flash memory blocks are to be locked and which blocks are not locked.

The records defined in step 2, determine the reset values of the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the following blocks are to be locked in the password 0 group:

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash[4]
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

From the PASS section of Table 9-6 in Device Configuration Format (DCF) Records :

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to Figure 36-5, the values for the DCF record are:

- Data[31:0] is 0x0030_0000 (bits 0–31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47–61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0 (see footnotes in Figure 36-6.

The following DCF record provides secure write protection for the blocks in Table 36-1 by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit No	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	x^1	x^2

Figure 36-6. Sample DCF record for secure write protection

The remaining blocks mapped to the PASS_LOCK0_PG*n* registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to flash blocks 4 and 5 set to '1'.

CAUTION

The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

See the PASS section of DCF client list table (Table 9-6 in Device Configuration Format (DCF) Records).

36.4.2 Overriding secure write protection

The "PG*n*" (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the flash module's SEL0–2 registers and LOCK0–2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PG*n* registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to '1' in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to '0' in any of the registers makes the corresponding block available for programming or erase.

If a bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to '0'. If the same bit is set to '1' in more than one of the PASS_LOCK*x*PG*n* register sets, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

Figure 36-7 shows the PASS_LOCK0_PG*n* register. As you can see, it has the same structure as the flash module's LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCK*x*_PG*n* registers and the flash LOCK*n* registers is how the reset values are determined.

By default, all mapped bits are set to '1', locking their corresponding flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

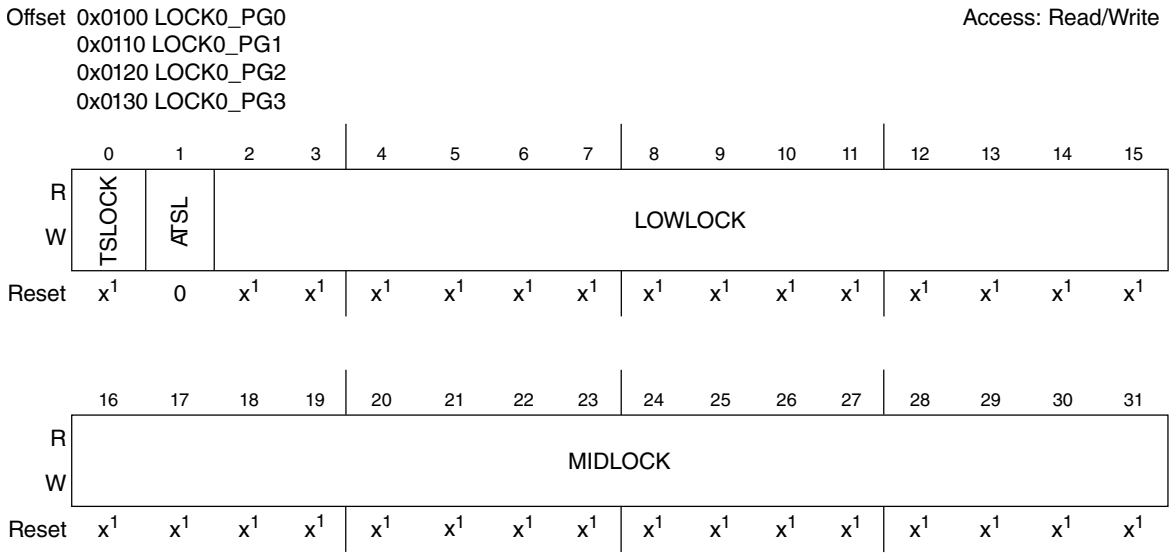


Figure 36-7. LOCK0_PG*n* register

Note

1. The values in the PASS_LOCK*x*PG*n* registers override the settings in the flash module's LOCK0–3 registers.
2. Values written to the PASS_LOCK*x*PG*n* registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection.
3. The PASS_LOCK*x*PG*n* bits take effect after the life cycle is matured to "OEM Production" or older.

Example: 36.4.2.1 Overriding secure write protection

Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCK*x*_PG*n* registers, so the reset value for each mapped bit defaults to '1', causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCK*x*_PG*n* registers override the settings in the flash module's LOCK0–3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module's LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to '0' to make the desired flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1–3.

Just as in previous examples, the result is the following blocks are available for program and erase operations but at a much higher level of security.

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash[4]
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code

36.5 Secure read protection

Read Protection is ability of the flash controller to block read accesses to the flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool

- DMA reads
- SIPI reads
- All other types of bus master reads

When secure read protection is active, the affected flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a flash read protected region terminates with a bus error.

Most instances of flash memory read protection in the MPC5777M microcontroller are *not* configurable by the customer. An example of this is the 256-bit passwords stored in UTEST flash memory. Password-based read protection *is* customer-configurable[4] and is similar to the password-based write protection mechanism detailed in Secure write protection.

There are two significant differences between password-based read protection and password-based write protection in the MPC5777M microcontroller.

1. Read protection can only be applied to pre-defined groups of flash memory blocks. These groups, called "Read Lock groups", cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to "Failure Analysis".

Another difference is that only the PASS_LOCK3_PG*n* register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

4. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

Note

The mappings of PASS_LOCK3_PG*n* register bits to flash blocks vary with different microcontrollers. See the flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the MPC5777M microcontroller.

The next sections discuss implementing secure read protection and overriding secure write protection.

36.5.1 Implementing secure read protection

Implementation of secure read protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) are stored in a protected OTP area of device flash named UTEST flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n* –PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.
2. DCF records must be created to define which flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PG*n* registers.

   Like the other PASS_LOCK*x*_PG*n* registers, the PASS_LOCK3_PG*n* registers have two functions.

   - Reading the registers provides a lock status for each Read Locking group.
   - Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PG*n* registers is shown in Figure 36-8. The fields relevant to implementing secure read protection are RL4–RL0.

Offset 0x010C LOCK3_PG0
0x011C LOCK3_PG1
0x012C LOCK3_PG2
0x013C LOCK3_PG3

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4	RL3	RL2	RL1	RL0
W																
Reset	x[1]	x[2]	x[2]	0	1	1	1	1	0	0	0	x[2]	x[2]	x[2]	x[2]	x[2]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256LCK_U															
W																
Reset	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]

Figure 36-8. PASS_LOCK3_PG*n* Register

Example: 36.5.1.1 Implementing secure read protection

Implementing secure read protection

Assume four passwords have been previously implemented.

Following is a example of a mapping of flash blocks to Read Locking groups.

Table 36-2. Sample low and mid address space flash block bit mapping

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x0040_4000	0x0040_7FFF	16	0	LOCK0	LOWLOCK[0]	16	—	0	BAF
0x00FC_0000	0x00FC_3FFF	16	0	LOCK0	LOWLOCK[1]	17	1	1	Code Flash
0x00FC_4000	0x00FC_7FFF	16	0	LOCK0	LOWLOCK[2]	18	1	2	Code Flash
0x00FC_8000	0x00FC_BFFF	16	1	LOCK0	LOWLOCK[3]	19	1	3	Code Flash
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash
0x0060_C000	0x0060_FFFF	16	1	LOCK0	LOWLOCK[5]	21	3	5	Secure Code
0x00FD_0000	0x00FD_7FFF	32	0	LOCK0	LOWLOCK[6]	22	1	6	Code Flash
0x00FD_8000	0x00FD_FFFF	32	1	LOCK0	LOWLOCK[7]	23	1	7	Code Flash
0x00FE_0000	0x00FE_FFFF	64	0	LOCK0	LOWLOCK[8]	24	1	8	Code Flash
0x00FF_0000	0x00FF_FFFF	64	0	LOCK0	LOWLOCK[9]	25	1	9	Code Flash
0x0061_0000	0x0061_FFFF	64	1	LOCK0	LOWLOCK[10]	26	3	10	Secure Code
0x0062_0000	0x0062_FFFF	64	1	LOCK0	LOWLOCK[11]	27	3	11	Secure Code

Table continues on the next page...

Table 36-2. Sample low and mid address space flash block bit mapping (continued)

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x0068_0000	0x0068_3FFF	16	2	LOCK0	MIDLOCK[0]	0	4	12	Secure Data
0x0068_4000	0x0068_7FFF	16	3	LOCK0	MIDLOCK[1]	1	4	13	Secure Data
0x0040_0000	0x0040_3FFF	16	0	LOCK0	TSLOCK	31	0	32	UTEST

To provide two level of password protection for only Read Lock group 3 and leave all other groups unlocked, the RL3 bit can be set in any two of the four PASS_LOCK3_PG*n* registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RL*n* bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of Table 9-6 in Device Configuration Format (DCF) Records.

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to Figure 36-5, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG1
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0111 (bits 47–61)

    - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
    - Preferred Stop value is 0.
- For LOCK3_PG2
    - Data[31:0] is 0x0000_0000 (bits 0–31)
    - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
    - Address[16:2] is 0b000_0000_0100_1011 (bits 47–61)
    - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
    - Preferred Stop value is 0.
- For LOCK3_PG3
    - Data[31:0] is 0x0000_0000 (bits 0–31)
    - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
    - Address[16:2] is 0b000_0000_0100_1111 (bits 47–61)
    - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
    - Preferred Stop value is 0.

36.5.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PG*n* registers (see Figure 36-8) contain five read protection fields, RP0–RP4, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG*n*[RL*n*] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to '1', causing the blocks associated with each read group to be read locked.

Example: 36.5.2.1 Overriding secure read protection

Overriding secure read protection

Building on Implementing secure read protection, where Read Lock group 3 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL3 bit to '0' in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL3] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

36.6 Debug port enable/disable

The mechanism for temporarily enabling the MPC5777M debug port after it has been locked by its life cycle state is similar to the flash memory read protection mechanism detailed in Secure read protection. The difference is that the PASS_LOCK3_PG*n*[DBL] field is used instead of the RP0–RP4 fields.

36.7 Tamper detection

The Tamper Detection Module (TDM) provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

This section provides details on the mapping between DCF record bits and individual flash memory blocks to be selected for tamper detection.

See Tamper Detection Module (TDM) for details on tamper detection, including the DCF clients and two additional memory-mapped registers.

There are two significant differences between selecting blocks for tamper detection and selecting blocks for other activities, e.g., a flash erase operation:

1. Selection of blocks for tamper detection protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. The selection of blocks for tamper detection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

The Tamper Detect section of DCF client list table (Table 9-6 in Device Configuration Format (DCF) Records) gives the CS and Address value combinations that correspond to groups of LOCK0–LOCK3 registers, similar to the flash module's LOCK0–LOCK3 registers.

The mapping of individual flash blocks to TDM DCF record data field bits is shown in included in the detailed flash block map in the flash section of the Device Configuration chapter.

36.7.1 Implementing tamper detection

Implementing tamper detection requires:

1. Creating the tamper detect diary
2. Assigning blocks to Tamper Detection Regions (TDRs)

The tasks are detailed in the following sections

36.7.2 Creating the tamper detect diary

Tamper detection in the MPC5777M microcontroller is user-defined. The basic functionality provided is an erase counter, or diary, for 6 user-defined regions, called "tamper detect regions", or TDRs, within the flash memory. Each TDR is defined using a set of DCF records that is similar in function and layout to the flash module's SEL*n* and LOCK*n* registers, which contain bits matched to specific flash blocks.

For a program or erase operation to be performed on a block included in a TDR, a record must be written to a "diary" that tracks flash modification operations. The format and size of records is customer-defined. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes.

Note

The limiting factor for tamper detection records is the diary size. The total diary size is given by the number of TDRs (6 for MPC5777M) multiplied by 2 KB, the portion of the diary allotted to records associated with each TDR.

Since the minimum flash programming operation size is 8 bytes, the maximum number of tamper detect records for a given TDR is 256 8-byte entries.

The diary can be placed within any flash block in the flash array but to maintain the security of the TDR, there are two constraints:

- The block where the diary is placed must be assigned as OTP within the OTP registers (covered in the next section)
- The diary base address must be on a 4 KB boundary, therefore the least significant 12 bits of the base address must all be '0'

Warning

The tamper detect diary base address DCF record is a "write once" record. All subsequent tamper detect diary base address DCF records created are ignored.

Example: 36.7.2.1 Creating the tamper detect diary

Creating the tamper detect diary

If the total space required for a diary is 12 KB (as in the case of 6 TDRs), any flash block can be used because the smallest flash blocks in the MPC5777M microcontroller are 16 KB. Since the diary base address must be on a 4 KB, boundary, though, if the diary base address is within a 16 KB flash block, it can only be assigned to the base address of the block or the address given by the base address plus 4 KB.

If the base address of the 16 KB block is 0x00FC4000, the base address of the diary is given by:

0b0000_0000_1111_1100_0000_0000_0000_0000 (16 KB base address)
or
0b0000_0000_1111_1100_0001_0000_0000_0000 (16 KB base address + 4 KB)

The base address of each TDR's section of the diary is:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

We need to create one DCF record that defines the base address for the diary. The information required is from the Tamper Detect section of Table 9-6 in Device Configuration Format (DCF) Records.

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0000	Diary Base Address

Referring to Figure 36-5, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x00FC4000 (the base address)
  - CS[14:0] is 0b000_0000_0001_0000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0000_0000 (bits 47–61)
  - Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
  - Preferred Stop value is 0.

See the Tamper Detect section of DCF client list table (Table 9-6 in Device Configuration Format (DCF) Records) for details.

Note

User software must check whether the diary is full. If the diary area for a TDR is full, user software must program a DCF record to override blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry. This is detailed in Overriding tamper detection.

36.7.3 Assigning blocks to Tamper Detection Regions (TDRs)

Tamper detection regions are defined by creating TDR*x*_LOCK*n* DCF records. For example, in a microcontroller with six tamper detection regions, the following records define the TDRs:

- TDR0_LOCK0, TDR0_LOCK1, TDR0_LOCK2, TDR0_LOCK3
- TDR1_LOCK0, TDR1_LOCK1, TDR1_LOCK2, TDR1_LOCK3
- TDR2_LOCK0, TDR2_LOCK1, TDR2_LOCK2, TDR2_LOCK3
- TDR3_LOCK0, TDR3_LOCK1, TDR3_LOCK2, TDR3_LOCK3
- TDR4_LOCK0, TDR4_LOCK1, TDR4_LOCK2, TDR4_LOCK3
- TDR5_LOCK0, TDR5_LOCK1, TDR5_LOCK2, TDR5_LOCK3

Each set of TDR*x*LOCK*n* DCF records mapped in exactly the same way as the flash module's SEL*n* and LOCK*n* registers (see the detailed flash block map in the flash section of the Device Configuration chapter for the mapping).

In the TDR*x*LOCK*n* DCF records:

- A bit set to '1' means the corresponding flash block is assigned to this TDR
- A bit set to '0' means the corresponding flash block is NOT assigned to this TDR

The default state (no TDRs defined) is defined as 'Block NOT assigned to the TDR'.

36.7.4 Overriding tamper detection

When a TDR's diary section is full, a Tamper Override (TO) record must be created before a block assigned to the TDR can be programmed or erased.

The details of the data section of the DCF record used to override tamper detection is shown in Figure 36-9.

warning

Bits in this DCF record is writable from 0 to 1only. Attempted writes of bits from 1 to 0 are ignored.

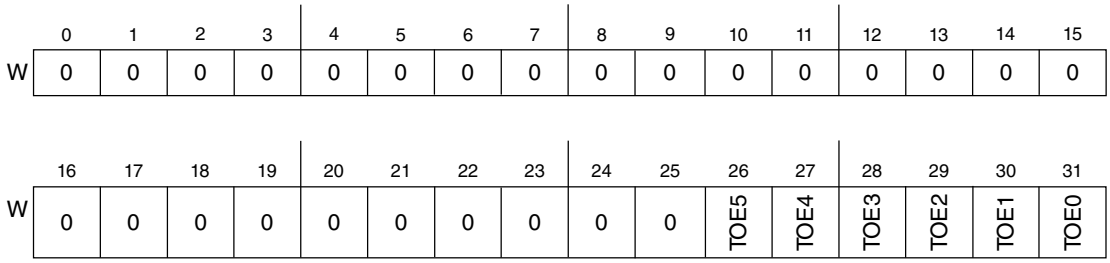


Figure 36-9. Data portion of Tamper Region Override (TO) DCF record

Example: 36.7.4.1 Implementing secure write protection

Implementing secure write protection

Assume the diary Tamper Detection Region 3 is full

We need to create one DCF record that defines that overrides tamper detection for the tamper detection region whose diary is full. The information required is from the Tamper Detect section of Table 9-6 in Device Configuration Format (DCF) Records.

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0001	Tamper Region Override

Referring to Figure 36-5, the values for the DCF record are:

- Data[31:0] is 0x0000_0008 (TOE3)
- CS[14:0] is 0b000_0000_0001_0000 (bits 32–46)
- Address[16:2] is 0b000_0000_0000_0001 (bits 47–61)
- Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
- Preferred Stop value is 0.

See the Tamper Detect section of DCF client list table (Table 9-6 in Device Configuration Format (DCF) Records) for details.

36.8 Implementing OTP

Any flash block within the flash array can be assigned, at any time, to be OTP. OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (i.e., flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

warning

After a flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in Table 9-6 in Device Configuration Format (DCF) Records. See the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

36.9 Implementing test mode disable

The MPC5777M flash memory module includes a mechanism to disable manufacturer entry into test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method are able to have possible failures analyzed by manufacturer's failure analysts.

Test mode disable prevents all high voltage operations to the flash executed by the internal state-machine, as well as reads through the state machine, and reads through the Array Integrity state machine when using test mode interfaces.

Optionally, the customer can create a password to enable manufacturer entry into test mode.

36.9.1 Unconditional test mode disable seal

Warning

This process is not reversible. After completing these steps the MPC5777M microcontroller is permanently prevented from entering test mode for factory analysis.

The steps for unconditionally disabling test mode are the same steps required for implementing passcode-protected entry into test mode with one exception—the passcode specified must be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To unconditionally disable test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST flash memory map.
2. Program one of the three *invalid* passcodes shown above into the Test Mode Override Passcode area shown in the UTEST flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTEST flash memory map.

After the next reset is asserted, Test Mode is permanently disabled.

Note

1. Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.
2. The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

36.9.2 Passcode-protected test mode disable seal

The steps for implementing passcode-protected entry into test mode are the same required for unconditionally disabling test mode with the exception that —the passcode specified must NOT be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To implement passcode-protected test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST flash memory map.
2. Program a valid passcode into the Test Mode Override Passcode area shown in the UTEST flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTEST flash memory map.

After the next reset is asserted, Test Mode is permanently passcode protected.

Note

1. Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.
2. The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

36.9.3 Selecting flash memory blocks for test mode disable seal

Selecting flash memory blocks for test mode disable seal protection is similar to selecting blocks for tamper detection in that it involves writing to non-register flash locations. In the case of the test mode disable seal function, there are two areas of UTEST flash memory that have identical mapping to flash memory blocks. When the test mode disable seal takes effect, the mapped bits are logically ORed to determine the blocks to be protected.

CAUTION

1. The mapping of bits to blocks for this function is different from the mapping used for other functions. See the table in the device configuration chapter for the map.
2. Selecting a block for test mode disable requires a '0' in the mapped bit—not a '1' as with other functions.

The mapping of individual flash blocks to Test Mode Disable Block Select bits is shown in Table 36-3.

Table 36-3. Mapping of Test Mode Disable Block Select bitsto flash blocks

Block	Byte Offset[-1]	Test Mode Disable Block Select Bit
32 KB Code Flash block1	0	0
32 KB Code Flash block0		1
16 KB HSM Code block5		2
16 KB Code Flash block4		3
16 KB Code Flash block3		4
16 KB Code Flash block2		5
16 KB Code Flash block1		6
16 KB BAF block0		7
—	1	8
—		9
—		10
—		11
64 KB HSM Code block3		12
64 KB HSM Code block2		13
64 KB Code Flash block1		14
64 KB Code Flash block0		15
—	2	16
—		17
—		18
—		19
—		20
—		21
16 KB HSM EEPROM block1		22
16 KB HSM EEPROM block0		23
—	3	24
—		25
—		26
—		27
—		28
—		29
—		30
—		31
—	4	32
—		33
—		34
—		35
64 KB EEPROM block3		36
64 KB EEPROM block2		37
64 KB EEPROM block1		38

Table continues on the next page...

Table 36-3. Mapping of Test Mode Disable Block Select bitsto flash blocks (continued)

Block	Byte Offset[-1]	Test Mode Disable Block Select Bit
64 KB EEPROM block0		39
—	5	40
—		41
—		42
—		43
—		44
—		45
—		46
—		47
—	6	48
256 KB Flash block7		49
256 KB Flash block6		50
256 KB Flash block5		51
256 KB Flash block4		52
256 KB Flash block3		53
256 KB Flash block2		54
256 KB Flash block1		55
256 KB Flash block0	7	56
—		57
—		58
256 KB Flash block13		59
256 KB Flash block12		60
256 KB Flash block11		61
256 KB Flash block10		62
256 KB Flash block9		63
—	8	64
—		65
—		66
—		67
—		68
—		69
—		70
—		71
—	9	72
—		73
—		74
—		75
—		76
—		77

Table continues on the next page...

Table 36-3. Mapping of Test Mode Disable Block Select bitsto flash blocks (continued)

Block	Byte Offset[-1]	Test Mode Disable Block Select Bit
—		78
—		79
—	10	80
—		81
—		82
—		83
—		84
—		85
—		86
—		87
—	11	88
—		89
—		90
—		91
—		92
—		93
—		94
—		95

1. The byte offset is from the start address of either the Test Mode Disable Block Select Group A or the start address of the Test Mode Disable Block Select Group B area of UTEST flash. See Table 2 in Memory Map for the locations.
2. The byte offset is from the start address of either the Test Mode Disable Block Select Group A or the start address of the Test Mode Disable Block Select Group B area of UTEST flash. See Table 2 in Memory Map for the locations.
3. The byte offset is from the start address of either the Test Mode Disable Block Select Group A or the start address of the Test Mode Disable Block Select Group B area of UTEST flash. See Table 2 in Memory Map for the locations.

36.10 Security configuration planning

At minimum, designing the flash security framework consists of selecting the security features to be implemented and determining the scope of implementation for each selected feature.

In devices containing a Hardware Security Module (HSM), significant planning must go into the implementation of HSM functions. The HSM is an independent software platform within the microcontroller. It is intended to be used to implement advanced security functions that are defined by customer software.

Note

Details of advanced security features, including the advanced security architecture and the Hardware Security Module (HSM) are contained in the *MPC5777M Microcontroller Security Reference Manual*. Distribution of the document is restricted to qualified customers.

The basic security features available in the MPC5777M microcontroller include:

- Secure write protection
- Secure read protection
- Debug port enable/disable
- One-time-programmable (OTP) flash memory block assignment
- Factory test mode disable

36.10.1 Hardware Security Module (HSM)

Details of the HSM are included in the *MPC5777M Microcontroller Security Reference Manual*. It is mentioned here because the flash memory module includes blocks that can be configured for exclusive access by the HSM. If the HSM is to be used in an application it is advisable to protect HSM code by configuring it for exclusive access by the HSM. The blocks configurable for exclusive access by the HSM appear in Figure 3-3 in Embedded Memories. They are labeled as secure code flash or secure EEPROM emulation blocks. They are not configured for HSM-exclusive access by default, though —they are available to act as secure flash areas but must be explicitly configured for that purpose. See the *MPC5777M Microcontroller Security Reference Manual* for details.

36.10.2 Creating password groups

Most basic security features in the MPC5777M microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTEST memory map (Table 5-6) in Memory Map.

The UTEST memory map fields to be written are:

- PASS Password Group 0

- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Warning

The password fields are in OTP UTEST flash, so they can only be written once.

36.10.3 Planning secure write protection

Up to four levels of password-secured write protection can be assigned to any block. Note that by default, all blocks of user flash are write protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured write protection. The write protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure write protection, please thoroughly review the implementation steps detailed in Implementing secure write protection and Table 9-6 in Device Configuration Format (DCF) Records.

36.10.4 Planning secure read protection

Up to four levels of password-secured write protection can be assigned to flash blocks in pre-defined Read Locking groups. Note that by default, all blocks of user flash are read protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured read protection. The read protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure read protection, please thoroughly review the implementation steps detailed in Implementing secure read protection and Table 9-6 in Device Configuration Format (DCF) Records.

36.10.5 Planning debug port enable/disable

Please see Debug port enable/disable and Table 9-6 in Device Configuration Format (DCF) Records.

36.10.6 Planning OTP flash memory block assignment

Any block of user flash can be configured as OTP. The need to configure blocks as OTP will depend on the data or code to be stored in the block. It is important to note that configuring a flash block as OTP is not reversible.

Before creating DCF records to implemented OTP flash blocks, please thoroughly review the implementation steps detailed in Implementing OTP and Table 9-6 in Device Configuration Format (DCF) Records.

36.10.7 Planning factory test mode disable

Please see Implementing test mode disable and Table 9-6 in Device Configuration Format (DCF) Records.

Chapter 37
External Bus Interface (EBI)

37.1 Introduction

This section contains an overview, list of features and modes of operation for the External Bus Interface (EBI).

37.1.1 Overview

The External Bus Interface (EBI) handles the transfer of information between the internal busses and the memories or peripherals in the external address space. The EBI includes a memory controller that generates interface signals to support a variety of external memories. This includes Single Data Rate (SDR) burst mode flash, SRAM, and asynchronous memories. It supports 3 regions (via chip selects), each with its own programmed attributes.

37.1.2 Features

- 32-Bit Address bus with transfer size indication
- 32-Bit Data bus (16-bit Data Bus Mode also supported)
- Memory controller with support for various memory types:
- synchronous burst SDR flash and SRAM
- asynchronous/legacy flash and SRAM
- Burst support (wrapped only)
- Port size configuration per chip select (16 or 32 bits)
- Support for Dynamic Calibration with up to 3 chip-selects

- Four Write/Byte Enable ($\overline{WE}$[0:3]/$\overline{BE}$ [0:3]) signals
- Slower-speed clock modes
- Stop and Module Disable Modes for power savings
- Optional automatic CLKOUT gating to save power and reduce EMI
- Misaligned access support
- Compatible with MPC5xx external bus (with some limitations)

37.1.3 Modes of Operation

The mode of the EBI is determined by the MDIS and DBM bits in the EBI_MCR, and the AD_MUX bit in the EBI_BR0-EBI_BR2. See Module Configuration Register (EBI_MCR) and Base Register Bank (EBI_BR*n*) for details. Slower-speed modes, Debug Mode, Stop Mode, and Factory Test Mode are modes that the MCU may enter, in parallel to the EBI being configured in one of its block-specific modes.

37.1.3.1 Single Master Mode

In Single Master Mode, the EBI responds to internal requests matching one of its regions, but ignores all externally-initiated bus requests. The MCU is the only master allowed to initiate transactions on the external bus in this mode; therefore, it acts as a parked master and does not have to arbitrate for the bus before starting each cycle. Single Master Mode is entered when MDIS=0 in the EBI_MCR.

37.1.3.2 Module Disable Mode

The Module Disable Mode is used for MCU power management. The clock to the non-memory-mapped logic in the EBI is stopped while in Module Disable Mode. Logic on the MCU external to the EBI is used to fully implement the Module Disable Mode (to shut off the clock). Module Disable Mode is entered when MDIS=1 in the EBI_MCR.

37.1.3.3 Stop Mode

When a request is made to enter Stop Mode (controlled in device logic outside EBI), the EBI block completes any pending bus transactions and acknowledges the stop request. After the acknowledgement, the system clock input may be shut off by the clock driver on the MCU. While the clocks are shut off, the EBI is not accessible.

37.1.3.4 EBI Clock Frequency

The maximum clock frequency for the EBI is one third the platform clock frequency, or 66.6 MHz, whichever is less. The EBI clock (CLKOUT) frequency is determined by a divider in the MC_CGM. See MC_CGM chapter for more information.

37.1.3.5 16-Bit Data Bus Mode

For MCUs that have only 16 data bus signals pinned out, or for systems where the use of a different multiplexed function (e.g. GPIO) is desired on 16 of the 32 data pins, the EBI supports a 16-bit Data Bus Mode. In this mode, only 16 data signals are used by the EBI. The user can select which 16 data signals are used (DATA[0:15] or DATA[16:31]) by writing the D16_31 bit in the EBI_MCR.

For EBI-mastered accesses, the operation in 16-bit Data Bus Mode (DBM=1, PS=x) is similar to a chip-select access to a 16-bit port in 32-bit Data Bus Mode (DBM=0, PS=1).

16-bit Data Bus Mode is entered when DBM=1 in the EBI_MCR.

Note

Since there is not an external TA pin, non-chip-select accesses are not supported.

37.1.3.6 Debug Mode

When the MCU is in Debug Mode, the EBI behavior is unaffected and remains dictated by the mode of the EBI.

37.2 External Signal Description

This section lists the external pins and describes the external signals of the EBI.

37.2.1 Overview

The following table lists the external pins used by the EBI. Not all signals listed here are available external to the chip.

Table 37-1. Signal Properties

Name	I/O Type	Function	Pull[1]
ADDR[8:31] ADDR[8:11]_WE[3:0]/BE[3:0]	I/O	Address bus	-
$\overline{\text{BDIP}}$	Output	Burst Data in Progress	Up
CLKOUT[2]	Output	Clockout	-
$\overline{\text{CS}}$[0:2]	Output	Chip Selects	-
DATA[0:31]	I/O	Data bus[3]	-
OE	Output	Output Enable	Up
RD_$\overline{\text{WR}}$	I/O	Read_Write	Up
TS	I/O	Transfer Start	Up

1. This column shows which signals require a weak pullup or pulldown. The EBI block does not contain these pullup/pulldown devices within the block. They are assumed to be in another module of the MCU (for example; pads module).
2. The CLKOUT signal is driven by the System Clock Block outside the EBI.
3. In Address/Data multiplexing modes, Data will also show the address during the address phase.

37.2.2 Detailed Signal Descriptions

This section describes the superset of signals for the EBI.

37.2.2.1 ADDR[8:31] — Address Lines 8-31

The ADDR[8:31] signals specify the physical address of the bus transaction.

The upper address lines ADDR[8:11] are shared with the WE[0:3]/BE[0:3] signals. The msb mapping is reversed to allow maximum address lines for a 16-bit memory, since only WE[0:1]/BE[0:1] are used in this case.

The address lines can be adjusted to the memory port size (8/16/32) via the EBI_OR[APS] bit on a per chip select basis. This allows a larger address space for wider memories.

Write enables are used to enable program operations to a particular memory. These signals can also be used as byte enables for read and write operation by setting the WEBS bit in the appropriate Base Register. $\overline{\text{WE}}$[0:3]/BE[0:3] are only asserted for chip-select accesses.

For chip-select accesses to a 16-bit port, only $\overline{\text{WE}}$ [0:1]/$\overline{\text{BE}}$[0:1] are used by the EBI, regardless of which half of the DATA bus is selected via the D16_31 bit in the EBI_MCR.

See Four Write/Byte Enable ($\overline{\text{WE}}$/$\overline{\text{BE}}$) Signals for more details on $\overline{\text{WE}}$[0:3]/$\overline{\text{BE}}$[0:3] functionality.

37.2.2.2 CLKOUT — Clockout

CLKOUT is a general-purpose clock output signal to connect to the clock input of SDR external memories.

37.2.2.3 $\overline{\text{CS}}$ [0:2] — Chip Selects 0-2

$\overline{\text{CS}}$x is asserted by the master to indicate that this transaction is targeted for a particular memory bank on the Primary external bus.

$\overline{\text{CS}}$ is driven in the same clock as the assertion of $\overline{\text{TS}}$ and valid address, and is kept valid until the cycle is terminated. See Memory Controller with Support for Various Memory Types for details on chip-select operation.The port size for the least significant address line, and the write/byte enable select is configurable for each chip select.

37.2.2.4 DATA [0:31] — Data Lines 0-31

The DATA[0:31] signals contain the data to be transferred for the current transaction.

DATA[0:31] is driven by the EBI when it owns the external bus and it initiates a write transaction to an external device.

DATA[0:31] is driven by an external device during a read transaction from the EBI.

For 8-bit and 16-bit transactions, the byte lanes not selected for the transfer do not supply valid data.

DATA[0:31] is driven by the EBI in the address phase with the ADDR value if the Address on Data multiplexing mode is enabled.

In 16-bit Data Bus Mode, (or for chip-select accesses to a 16-bit port), only DATA[0:15] or DATA[16:31] are used by the EBI, depending on the setting of the D16_31 bit in the EBI_MCR. See 16-Bit Data Bus Mode.

37.2.2.5 $\overline{\text{OE}}$ — Output Enable

$\overline{\text{OE}}$ is used to indicate when an external memory is permitted to drive back read data. External memories must have their data output buffers off when $\overline{\text{OE}}$ is negated. $\overline{\text{OE}}$ is only asserted for chip-select accesses.

For read cycles, $\overline{\text{OE}}$ is asserted one clock after $\overline{\text{TS}}$ assertion (by default with EBI_BR[EOE]=0b00) and held until the termination of the transfer. For write cycles, $\overline{\text{OE}}$ is negated throughout the cycle.

37.2.2.6 RD_$\overline{\text{WR}}$ — Read / Write

RD_$\overline{\text{WR}}$ indicates whether the current transaction is a read access or a write access.

RD_$\overline{\text{WR}}$ is driven in the same clock as the assertion of $\overline{\text{TS}}$ and valid address, and is kept valid until the cycle is terminated.

37.2.2.7 $\overline{\text{TS}}$ — Transfer Start

$\overline{\text{TS}}$ is asserted by the current bus owner to indicate the start of a transaction on the external bus.

$\overline{\text{TS}}$ is only asserted for the first clock cycle of the transaction, and is negated in the successive clock cycles until the end of the transaction.

37.2.2.8 $\overline{\text{WE}}$ [0:3] / $\overline{\text{BE}}$ [0:3] — Write/Byte Enables 0-3

Write enables are used to enable program operations to a particular memory. These signals can also be used as byte enables for read and write operation by setting the WEBS bit in the appropriate Base Register. $\overline{\text{WE}}$[0:3]/BE[0:3] are only asserted for chip-select accesses.

For chip-select accesses to a 16-bit port, only $\overline{\text{WE}}$ [0:1]/$\overline{\text{BE}}$[0:1] are used by the EBI, regardless of which half of the DATA bus is selected via the D16_31 bit in the EBI_MCR.

See Four Write/Byte Enable ($\overline{WE}$/$\overline{BE}$) Signals for more details on $\overline{WE}$[0:3]/$\overline{BE}$[0:3] functionality.

37.3 Memory Map/Register Definition

NOTE

The EBI registers must not be written while a transaction to the EBI (from internal or external master) is in progress (or within 2 CLKOUT cycles after a transaction has just completed, to allow internal state machines to go IDLE) In those cases, the behavior is undefined.

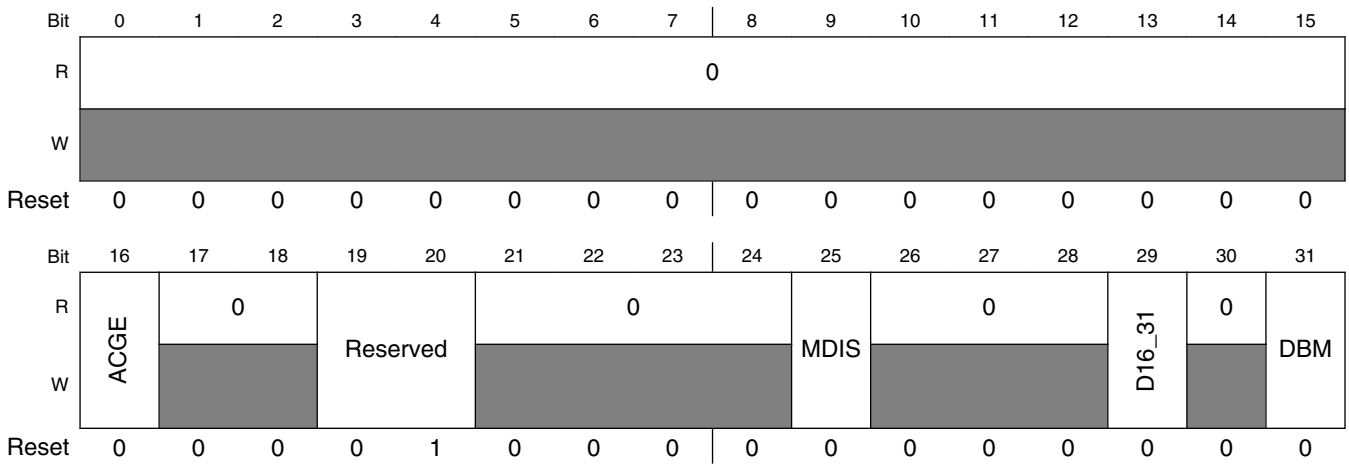
EBI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (EBI_MCR)	32	R/W	0000_0800h	37.3.1/1410
10	Base Register Bank (EBI_BR0)	32	R/W	2000_0002h	37.3.2/1411
14	Option Register Bank (EBI_OR0)	32	R/W	E000_0000h	37.3.3/1413
18	Base Register Bank (EBI_BR1)	32	R/W	2000_0002h	37.3.2/1411
1C	Option Register Bank (EBI_OR1)	32	R/W	E000_0000h	37.3.3/1413
20	Base Register Bank (EBI_BR2)	32	R/W	2000_0002h	37.3.2/1411
24	Option Register Bank (EBI_OR2)	32	R/W	E000_0000h	37.3.3/1413

37.3.1 Module Configuration Register (EBI_MCR)

The EBI Module Configuration Register contains bits which configure various attributes associated with EBI operation.

Address: 0h base + 0h offset = 0h



EBI_MCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ACGE	Automatic CLKOUT Gating Enable The ACGE bit enables the EBI feature of turning off CLKOUT (holding it high) during idle periods in-between external bus accesses. 0 Automatic CLKOUT Gating is disabled 1 Automatic CLKOUT Gating is enabled
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–20 Reserved	This field is reserved.
21–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 MDIS	Module Disable Mode The MDIS bit controls an internal EBI "enable clk" signal which can be used (if MCU logic supports) to control the clocks to the EBI. The MDIS bit allows the clock to be stopped to the non-memory mapped logic in the EBI, effectively putting the EBI in a software controlled power-saving state. See Module Disable Mode for more information. No external bus accesses can be performed when the EBI is in Module Disable Mode (MDIS=1). 0 Module Disable Mode is inactive (assert "enable clk" signal) 1 Module Disable Mode is active (negate "enable clk" signal)

Table continues on the next page...

EBI_MCR field descriptions (continued)

Field	Description
26–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 D16_31	Data Bus 16_31 Select The D16_31 bit controls whether the EBI uses the DATA[0:15] or DATA[16:31] signals, when in 16-bit Data Bus Mode (DBM=1) or for chip-select accesses to a 16-bit port (PS=1). **NOTE:** There is one usage case where D16_31=0 cannot be used. For systems that are using both a 32-bit non-chip-select device AND a 16-bit device (chip-select or non-chip-select), the user must set D16_31=1. Otherwise (if D16_31=0 is used), when an access to the 16-bit device is performed, the 32-bit non-chip-select device may decode the wrong upper address bits and may respond to that access, causing contention on the shared address/data lines going to both devices. This is due to the 32-bit device expecting address 8:15 on DATA[8:15] pins, whereas D16_31=0 has address 24:31 on those pins. This contention case is avoided when D16_31=1, because the shared address/data pins have the same functions for both the 16-bit and 32-bit memory accesses (address 16:31 on DATA[16:31] pins). This contention case is also avoided when only chip-select devices are used, since the devices ignore accesses when their corresponding chip-select is not asserted. 0 DATA[0:15] signals are used for 16-bit port accesses 1 DATA[16:31] signals are used for 16-bit port accesses
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 DBM	Data Bus Mode The DBM bit controls whether the EBI is in 32-bit or 16-bit Data Bus Mode. 0 32-bit Data Bus Mode is used 1 16-bit Data Bus Mode is used

37.3.2 Base Register Bank (EBI_BR*n*)

The EBI Base Registers are used to define the base address and other attributes for the corresponding chip select.

NOTE

The upper bits of the BA field [0:2] are tied to a fixed value of "001", in which case the reset value is this fixed value and not zero.

Address: 0h base + 10h offset + (8d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BA															
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BA	0		LWRN	PS	EOE		SBL	AD_MUX	BL	WEBS	TBDIP	GCSN	0	BI	V
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

EBI_BR*n* field descriptions

Field	Description
0–16 BA	Base Address These bits are compared to the corresponding unmasked address signals among ADDR[0:16] of the internal address bus to determine if a memory bank controlled by the memory controller is being accessed by an internal bus master. **NOTE:** An MCU has some of the upper bits of the BA field [0:2] tied to a fixed value of "001" internally in order to restrict the address range of the EBI for that MCU. Tied-off bits can be read but not written. These bits are ignored by the EBI during the chip-select address comparison. However, the internal bridge of the MCU most likely requires that the chip-select banks be located in memory regions corresponding to the fixed values chosen.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 LWRN	Late RD_$\overline{WR}$ Negation The LWRN bit determines the timing of RD_WR signal negation for a write transfer. 0 Negate RD_$\overline{WR}$ one cycle earlier than $\overline{CS}$ negation. See Figure 37-9. 1 Negate RD_$\overline{WR}$ the same cycle as $\overline{CS}$ negation. See Figure 37-12.
20 PS	Port Size The PS bit determines the data bus width of transactions to this chip-select bank. **NOTE:** In the case where the DBM bit in EBI_MCR is set for 16-bit Data Bus Mode, the PS bit value is ignored and is always treated as a '1'(16-bit port). 0 32-bit port 1 16-bit port
21–22 EOE	Early $\overline{OE}$ The EOE field determines the timing of $\overline{OE}$ signal assertion for a read transfer. When EBI_BR[ADMUX]=1, the EOE value is ignored and treated as 0b00 (in order to avoid contention on shared address/data bus for muxed transfers).
23 SBL	Short Burst Length The SBL bit provides support for a 2-word external burst (see Figure 37-31 in Small Access Example #5: 32-byte Read to 32-bit Port with SBL=1. 0 When SBL=0, the number of beats in a burst is determined by the BL bit. See Table 37-2 under BL bit description for SBL and BL encodings. 1 When SBL=1, the number of beats in a burst is automatically determined by the EBI to be 2 or 4 according to the Port Size (PS bit), regardless of BL bit value.
24 AD_MUX	Address on Data Bus Multiplexing The AD_MUX bit controls whether accesses for this chip select have the address driven on the data bus in the address phase of a cycle.

Table continues on the next page...

EBI_BR*n* field descriptions (continued)

Field	Description
	0 Address on Data Multiplexing Mode is disabled for this chip select. 1 Address on Data Multiplexing Mode is enabled for this chip select.
25 BL	Burst Length The BL bit (along with SBL bit) determines the amount of data transferredin a burst for this chip select, measured in 32-bit words. 0 When SBL=0, the number of beats in a burst is automatically determined by the EBI to be 4, 8, or 16 according tothe Port Size (PS bit) so that the burst fetches the number of words chosen by BL. For internal AMBA data bus width of 32-bits, the BL bit is ignored (treated as 1). 1 When SBL=1, the BL bit value is a don't care. See Table 37-2 for SBL and BL encodings.
26 WEBS	Write Enable / Byte Select This bit controls the functionality of the $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals. 0 The $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals function as $\overline{WE}$[0:3] 1 The $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals function as BE[0:3]
27 TBDIP	This bit determines how long the BDIP signal is asserted for each data beat in a burst cycle. 0 Assert $\overline{BDIP}$ throughout the burst cycle, regardless of wait state configuration 1 Only assert $\overline{BDIP}$ (BSCY+1) external cycles before expecting subsequent burst data beats
28 GCSN	Guarantee $\overline{CS}$ Negation The GCSN bit allows support for guaranteeing that the EBI will negate $\overline{CS}$ between all back-to-back transfer cases (even those that are part of a set of Small Accesses). See Figure 37-17. 0 Default operation ($\overline{CS}$ may or may not negate between transfers, depending on the particular back-to-back case). 1 Negate $\overline{CS}$ between all external transfers. This adds an extra dead cycle for some back-to-back cases.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 BI	Burst Inhibit This bit determines whether or not burst read accesses are allowed for this chip-select bank. 0 Enable burst accesses for this bank 1 Disable burst accesses for this bank. This is the default value out of reset.
31 V	Valid bit The user writes this bit to indicate that the contents of this Base Register and Option Register pair are valid. The appropriate $\overline{CS}$ signal does not assert unless the corresponding V-bit is set. 0 This bank is not valid 1 This bank is valid

37.3.3 Option Register Bank (EBI_OR*n*)

The EBI Option Registers are used to define the address mask and other attributes for the corresponding chip select.

NOTE

Some upper bits [0:2] of the AM field are tied to a fixed value of "111", in which case the reset value is this fixed value and not zero. Tied-off bits can be read but not written. See the corresponding Note for the Base Register BA field for more details.

Address: 0h base + 14h offset + (8d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AM															
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AM	0		APS	0	AWE	0		SCY				0	BSCY		0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EBI_OR*n* field descriptions

Field	Description
0–16 AM	Address Mask This field allows masking of any corresponding bits in the associated Base Register. Masking the address independently allows external devices of different size address ranges to be used. Any clear bit masks the corresponding address bit. . Any set bit causes the corresponding address bit to be used in comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at any time. **NOTE:** An MCU has some of the upper bits [0:2] of the AM field tied to a fixed value of "111" internally in order to restrict the address range of the EBI for that MCU. See the corresponding Note for the Base Register BA field for more details. Tied-off bits can be read but not written.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 APS	Address by Port Size The user writes this bit to cause the EBI to shift the address it drives out for accesses to this chip-select, such that the lowest address bit addresses an element of data that matches the port size specified in the corresponding EBI_BR[PS] field. APS=0, PS=X Default Byte Addressing APS=1, PS=0 Word Addressing: EBI drives internal address 6:29 to ADDR[8:31] APS=1, PS=1 Half-Word Addressing: EBI drives internal address 7:30 to ADDR[8:31] See Address by Port Size for additional details.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AWE	Address / Write Enable Select The user writes this bit to cause the EBI to drive its internal WE[0:3] onto the upper 4 address bus signals (such as in systems where upper 4 address bus signals are not needed and separate WE[0;3] pins are not available). AWE has no effect on the other address bits besides the upper 4. For example, for an SoC using a 24-bit address bus, the upper 4 bits are ADDR[8:11]. See Address / Write Enable Multiplexing for additional details.

Table continues on the next page...

EBI_OR*n* field descriptions (continued)

Field	Description
	0 Default address function driven on 4 upper address bus signals 1 WE[0:3] driven into 4 upper address bus signals
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 SCY	Cycle length in clocks This field represents the number of wait states (external cycles) inserted after the address phase in the single transfer case, or in the first beat of a burst, when the memory controller handles the external memory access. Values range from 0 to 15. This is the main parameter for determining the length of the cycle. The total cycle length for the first beat (including the $\overline{TS}$ cycle) = (2+SCY) external clock cycles . See Example Wait State Calculation for related application information.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 BSCY	Burst beats length in clocks This field determines the number of wait states (external cycles) inserted in all burst beats except the first, when the memory controller starts handling the external memory access and thus is using SCY[0:3] to determine the length of the first beat. The total memory access length for each beat is (1 + BSCY) external clock cycles. The total cycle length (including the TS cycle) = (2+SCY) + (#beats-1) * (BSCY+1). #beats is the number of beats (4,8,16) determined by BL and PS bits in Base Register. 00 0 - clock cycle wait states (1 clock per data beat) 01 1 - clock cycle wait states (2 clocks per data beat) 10 2 - clock cycle wait states (3 clocks per data beat) 11 0 - clock cycle wait states (4 clocks per data beat)
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

37.4 Functional Description

The following sections describe the EBI functionality.

37.4.1 External Bus Interface Features

37.4.1.1 32-Bit Data Bus (16-bit Data Bus Mode also supported)

The entire 32-bit data bus is available for external memory accesses. There is also a 16-bit Data Bus Mode available via the DBM bit in EBI_MCR. See 16-Bit Data Bus Mode.

37.4.1.2 Memory Controller with Support for Various Memory Types

The EBI contains a memory controller that supports a variety of memory types, including synchronous burst mode flash and SRAM, and asynchronous/legacy flash and SRAM with a compatible interface.

Each $\overline{CS}$ bank is configured via its own pair of Base and Option Registers. Each time an internal to external bus cycle access is requested, the internal address is compared with the base address of each valid Base Register (with 17 bits having mask). See Figure 37-1. If a match is found, the attributes defined for this bank in its BR and OR are used to control the memory access. If a match is found in more than one bank, the lowest bank matched handles the memory access (for example; bank 0 is selected over bank 1).

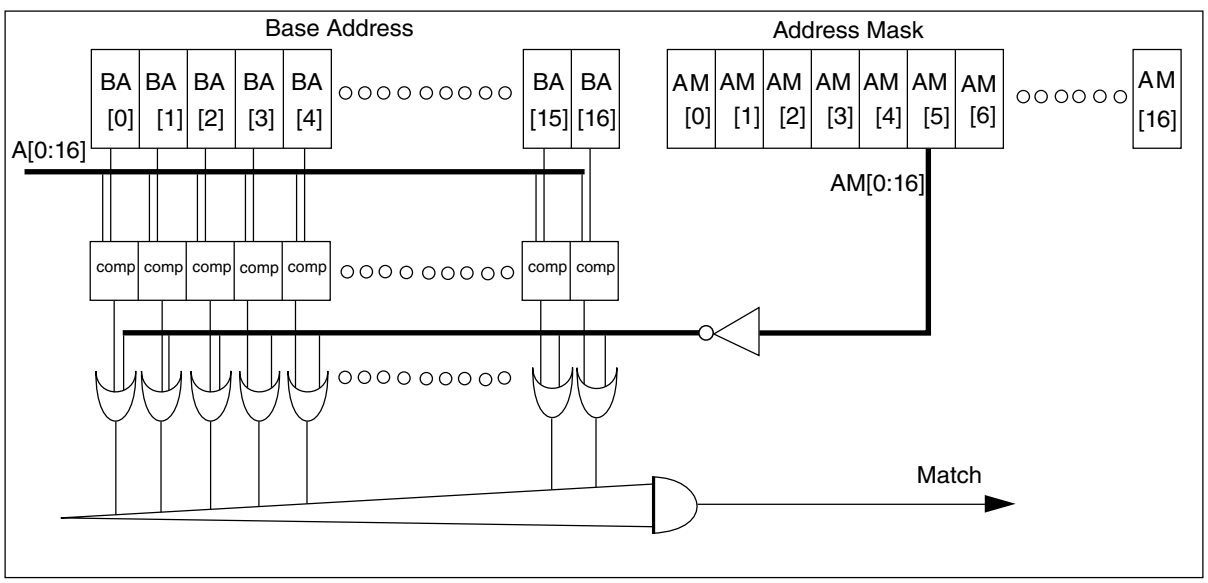


Figure 37-1. Bank Base Address and Match Structure

When a match is found on one of the chip-select banks, all its attributes (from the appropriate Base and Option Registers) are selected for the functional operation of the external memory access, such as:

- Number of wait states for a single memory access, and for any beat in a burst access
- Burst enable
- Port size for the external accessed device

See Module Configuration Register (EBI_MCR) and Base Register Bank (EBI_BR*n*) for a full description of all chip-select attributes.

When no match is found on any of the chip-select banks, the default transfer attributes shown in Table 37-2 are used.

Table 37-2. Default Attributes for Non-Chip-Select Transfers

CS Attribute	Default Value	Comment
PS	0	32-bit port size
BL	0	burst length is don't care since burst is disabled
WEBS	0	write enables
BI	1	burst inhibited
SCY	0	Cycle length in clocks
BSCY	0	Burst beats length in clocks

37.4.1.3 Burst Support (wrapped only)

The EBI supports burst read accesses of external burstable memory. To enable bursts to a particular memory region, clear the BI (Burst Inhibit) bit in the appropriate Base Register. External burst lengths of 2, 4 and 8 words are supported. Burst length is configured for each chip select by using the BL and SBL bits in the appropriate Base Register. See Burst Transfer for more details on burst operation.

Burst writes are not supported. Internal requests to write >32 bits (such as a cache line) externally are broken up into separate 32-bit or 16-bit external transactions according to the port size. See Small Accesses (Small Port Size and Short Burst Length) for more detail on these cases.

37.4.1.4 Port Size Configuration per Chip Select (16 or 32 bits)

The EBI supports memories with data widths of 16 or 32 bits. The port size for a particular chip select is configured by writing the PS bit in the corresponding Base Register.

37.4.1.5 Configurable Wait States

From 0 to 15 wait states can be programmed for any cycle that the memory controller generates, via the SCY bits in the appropriate Option Register. From 0 to 3 wait states between burst beats can be programmed using the BSCY bits in the appropriate Option Register.

37.4.1.6 Four Write/Byte Enable ($\overline{WE}$/$\overline{BE}$) Signals

The functionality of the $\overline{WE}$[0:3]/$\overline{BE}$ [0:3] signals depends on the value of the WEBS bit in the corresponding Base Register. Setting WEBS to 1 configures these pins as $\overline{BE}$[0:3], while resetting it to 0 configures them as $\overline{WE}$ [0:3]. The WE[0:3]/BE[0:3] signals are mapped to the address lines and are independently enabled for each chip select. $\overline{WE}$[0:3] are asserted only during write accesses, while $\overline{BE}$ [0:3] is asserted for both read and write accesses. The timing of the $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals remains the same in either case.

The upper Write/Byte Enable ($\overline{WE0}$/$\overline{BE0}$) indicates that the upper eight bits of the data bus (DATA[0:7]) contain valid data during a write/read cycle. The upper middle Write/Byte Enable ($\overline{WE1}$/$\overline{BE1}$) indicates that the upper middle eight bits of the data bus (DATA[8:15]) contain valid data during a write/read cycle. The lower middle Write/Byte Enable ($\overline{WE2}$/$\overline{BE2}$) indicates that the lower middle eight bits of the data bus (DATA[16:23]) contain valid data during a write/read cycle. The lower Write/Byte Enable ($\overline{WE3}$/$\overline{BE3}$) indicates that the lower eight bits of the data bus (DATA[24:31]) contain valid data during a write/read cycle.

Note

The exception to the preceding $\overline{WE}$/$\overline{BE}$ description is that for 16-bit port transfers (DBM=1 or PS=1), only the $\overline{WE}$[0:1]/$\overline{BE}$[0:1] signals are used, regardless of whether DATA[0:15] or DATA[16:31] are selected (via the D16_31 bit in the EBI_MCR). This means for the case where DATA[16:31] are selected, that $\overline{WE0}$ indicates that DATA[16:23] contains valid data, and $\overline{WE1}$ indicates that DATA[24:31] contains valid data.

The Write/Byte Enable lines affected in a transaction for a 32-bit port (PS = 0) and a 16-bit port (PS=1) are shown in Table 37-3. Only Big Endian byte ordering is supported by the EBI.

The following table applies to aligned internal master transfers only. In the case of a misaligned internal master transfer that is split into multiple aligned external transfers, not all of the write enables X'd in the table will necessarily assert. See Misaligned Access Support.

Table 37-3. Write/Byte Enable Signals Function

Transfer Size	EBI_MCR[SIZE]	Address		32-Bit Port Size				16-Bit Port Size[1]			
		A30	A31	$\overline{WE0}$/ $\overline{BE0}$	$\overline{WE1}$/ $\overline{BE1}$	$\overline{WE2}$ /$\overline{BE2}$	$\overline{WE3}$ /$\overline{BE3}$	$\overline{WE0}$ /$\overline{BE0}$	$\overline{WE1}$/ $\overline{BE1}$	$\overline{WE2}$ /$\overline{BE2}$	$\overline{WE3}$ /$\overline{BE3}$
Byte	01	0	0	X				X			

Table continues on the next page...

Table 37-3. Write/Byte Enable Signals Function (continued)

Transfer Size	EBI_MCR[SIZE]	Address		32-Bit Port Size				16-Bit Port Size[1]			
		A30	A31	WE0/ BE0	WE1/ BE1	WE2 /BE2	WE3 /BE3	WE0 /BE0	WE1/ BE1	WE2 /BE2	WE3 /BE3
	01	0	1		X				X		
	01	1	0			X		X			
	01	1	1				X		X		
16-bit	10	0	0	X	X			X	X		
	10	1	0			X	X	X	X		
32-bit	00	0	0	X	X	X	X	X[2]	X[2]		
Burst	00	0	0	X	X	X	X	X	X		

1. Also applies when DBM=1 for 16-bit data bus mode.
2. This case consists of two 16-bit external transactions, but for both transactions the $\overline{WE}$[0:1]/$\overline{BE}$[0:1] signals are the only $\overline{WE}$/$\overline{BE}$ signals affected.

37.4.1.7 Slower-Speed Clock Modes

For memories that cannot run with a full-speed external bus, the EBI supports slower-speed clock modes. See MC_CGM chapter for adjusting the CLKOUT and external bus frequency.

37.4.1.8 Stop and Module Disable Modes for Power Savings

See Modes of Operation for a description of the power saving modes.

37.4.1.9 Optional Automatic CLKOUT Gating

The EBI has the ability to hold the external CLKOUT pin high when the EBI's internal master state machine is idle and no requests are pending. The EBI outputs a signal to the pads logic in the MCU to disable CLKOUT. This feature is disabled out of reset, and can be enabled or disabled by the ACGE bit in the EBI_MCR.

Note

This feature must be disabled for multi-master systems. In those cases, one master is getting its clock source from the other master and needs it to stay valid continuously.

37.4.1.10 Misaligned Access Support

The EBI has limited misaligned access support. Misaligned non-burst chip-select transfers from internal masters are supported. The EBI aligns the accesses when it sends them out to the external bus (splitting them into multiple aligned accesses if necessary), so that external devices are not required to support misaligned accesses. Burst accesses (internal master) must match the internal bus size (64-bit aligned). See Misaligned Access Support for more details.

37.4.2 External Bus Operations

The following sections provide a functional description of the external bus, the bus cycles provided for data transfer operations and error conditions.

37.4.2.1 External Clocking

The CLKOUT signal sets the frequency of operation for the bus interface directly. Internally, the MCU uses a phase-locked loop (PLL) circuit to generate a master clock for all of the MCU circuitry (including the EBI) that is phase-locked to the CLKOUT signal. In general, all signals for the EBI are specified with respect to the rising-edge of the CLKOUT signal, and they are guaranteed to be sampled as inputs or changed as outputs with respect to that edge.

37.4.2.2 Reset

Upon detection of internal reset assertion, the EBI immediately ends all transactions (abruptly, not through normal termination protocol), and ignores any transaction requests that take place while reset is asserted.

37.4.2.3 Basic Transfer Protocol

The basic transfer protocol defines the sequence of actions that must occur on the external bus to perform a complete bus transaction. A simplified scheme of the basic transfer protocol is shown in Figure 37-2.

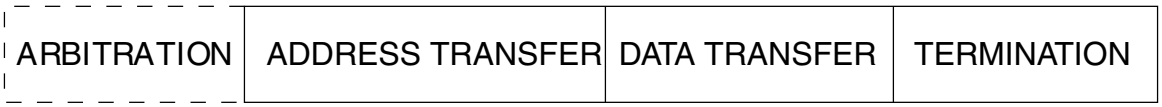


Figure 37-2. Basic Transfer Protocol

The arbitration phase is where bus ownership is requested and granted. This phase is not needed in Single Master Mode because the EBI is the permanent bus owner in this mode.

The address transfer phase specifies the address for the transaction and the transfer attributes that describe the transaction. The signals related to the address transfer phase are $\overline{TS}$, ADDR (or DATA if Address/Data multiplexing is used), $\overline{CS}$ [0:2] and RD_$\overline{WR}$. The address and its related signals (with the exception of $\overline{TS}$) are driven on the bus with the assertion of the $\overline{TS}$ signal, and kept valid until the bus master receives $\overline{TA}$ asserted (the EBI holds them one cycle beyond $\overline{TA}$ for writes). Writes with internal $\overline{TA}$, RD_$\overline{WR}$ are not held one cycle past $\overline{TA}$.

The data transfer phase performs the transfer of data, from master to slave (in write cycles) or from slave to master (on read cycles), if any is to be transferred. The data phase may transfer a single beat of data (1-4 bytes) for non-burst operations or a 2-beat, 4-beat, 8-beat, or 16-beat burst of data (2 or 4 bytes per beat depending on Port Size) when burst is enabled. On a write cycle, the master must not drive write data until after the address transfer phase is complete. This is to avoid electrical contentions when switching between drivers. The master must start driving write data one cycle after the address transfer cycle. The master can stop driving the data bus as soon as it samples the $\overline{TA}$ line asserted on the rising edge of CLKOUT. To facilitate asynchronous write support, the EBI keeps driving valid write data on the data bus until 1 clock after the rising edge where RD_$\overline{WR}$ and $\overline{WE}$ are negated. See Figure 37-9 for an example of write timing. On a read cycle, the master accepts the data bus contents as valid on the rising edge of the CLKOUT in which the $\overline{TA}$ signal is sampled asserted. See Figure 37-4 for an example of read timing.

The termination phase is where the cycle is terminated according to the values in the EBI Base Registers. Termination is discussed in detail in Termination Signals Protocol.

Note

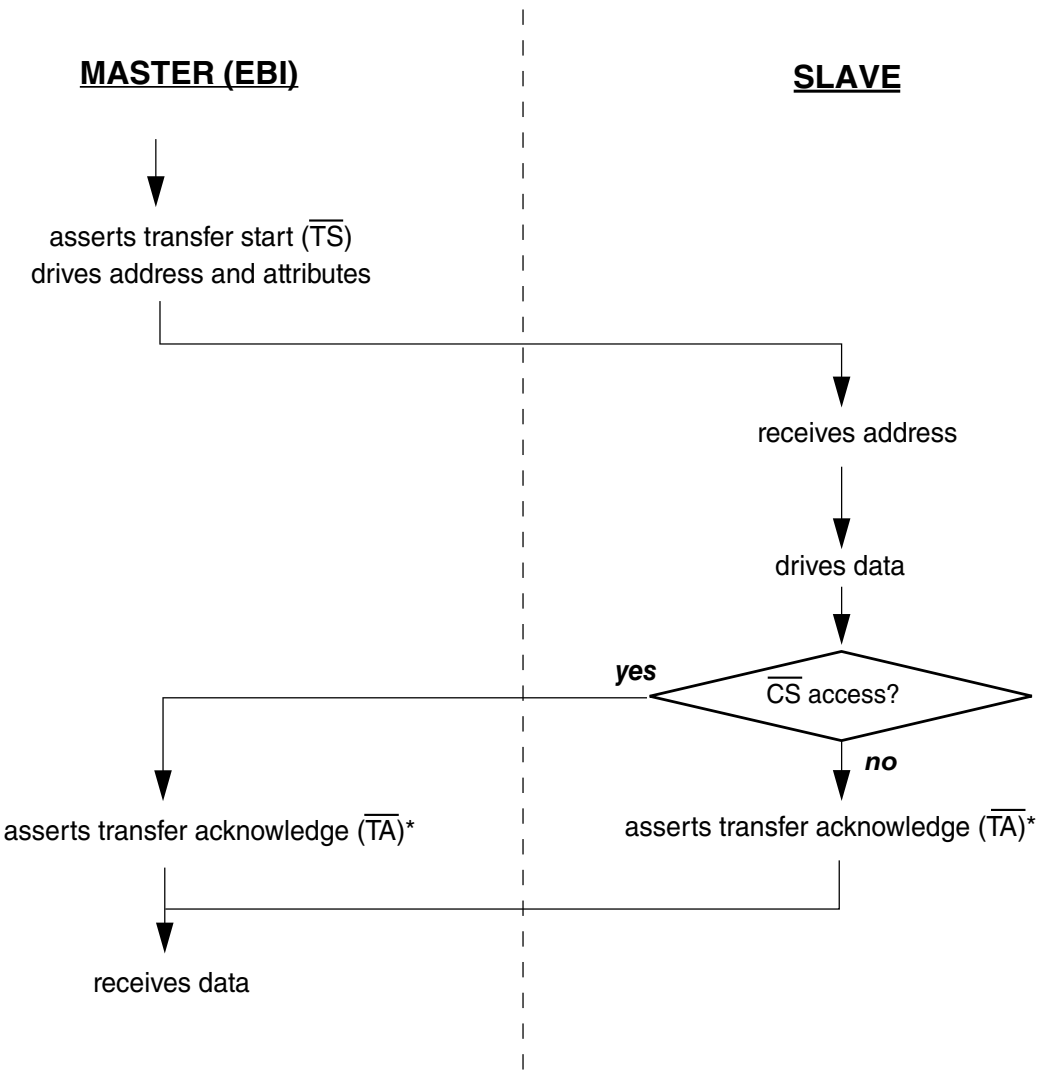
In the timing diagrams in this document, asynchronous relationships between signals that switch in the same CLKOUT cycle are not guaranteed. For example, in Figure 37-9, $\overline{WE}$ and write DATA change during the same CLKOUT cycle. There is no guarantee that DATA will be stable before $\overline{WE}$ assertion. External devices should not be latching write DATA on $\overline{WE}$ assertion, but instead must use a signal edge that takes place in a later CLKOUT cycle, such as $\overline{WE}$ negation.

37.4.2.4 Single Beat Transfer

The flow and timing diagrams in this section assume that the EBI is configured in Single Master Mode. Therefore, arbitration is not needed and is not shown in these diagrams

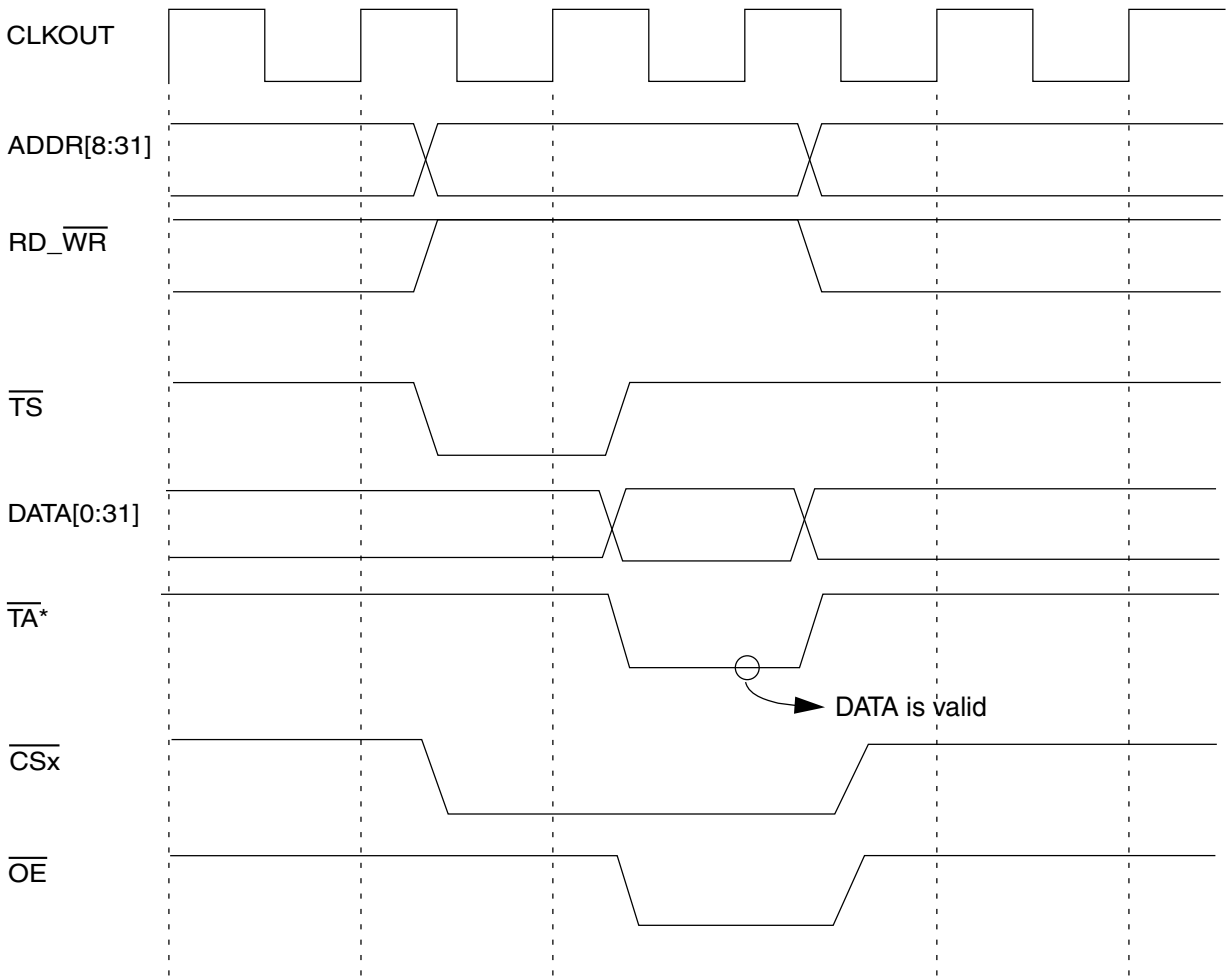
37.4.2.4.1 Single Beat Read Flow

The handshakes for a single beat read cycle are illustrated in the following flow and timing diagrams.



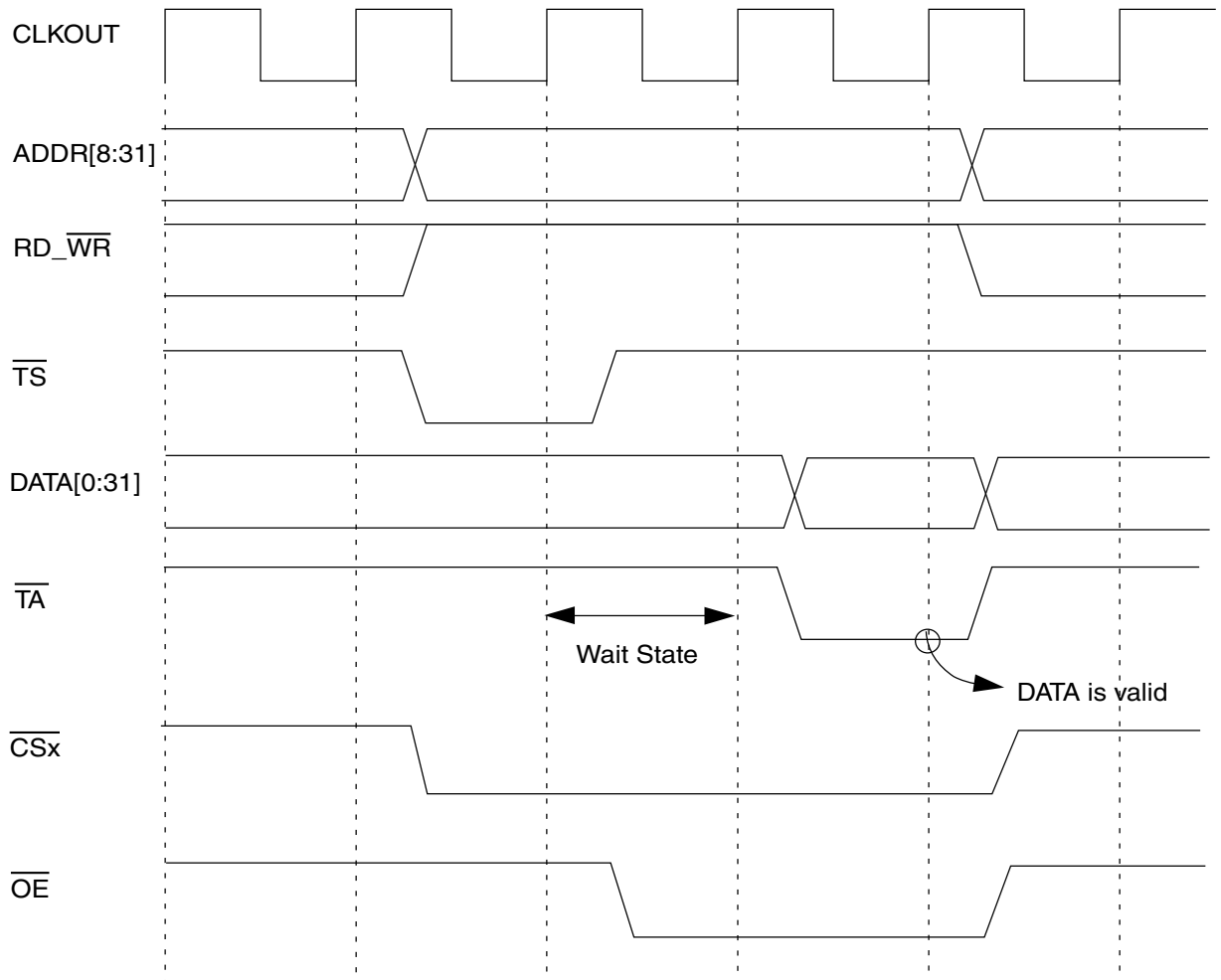
* There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-3. Basic Flow Diagram of a Single Beat Read Cycle



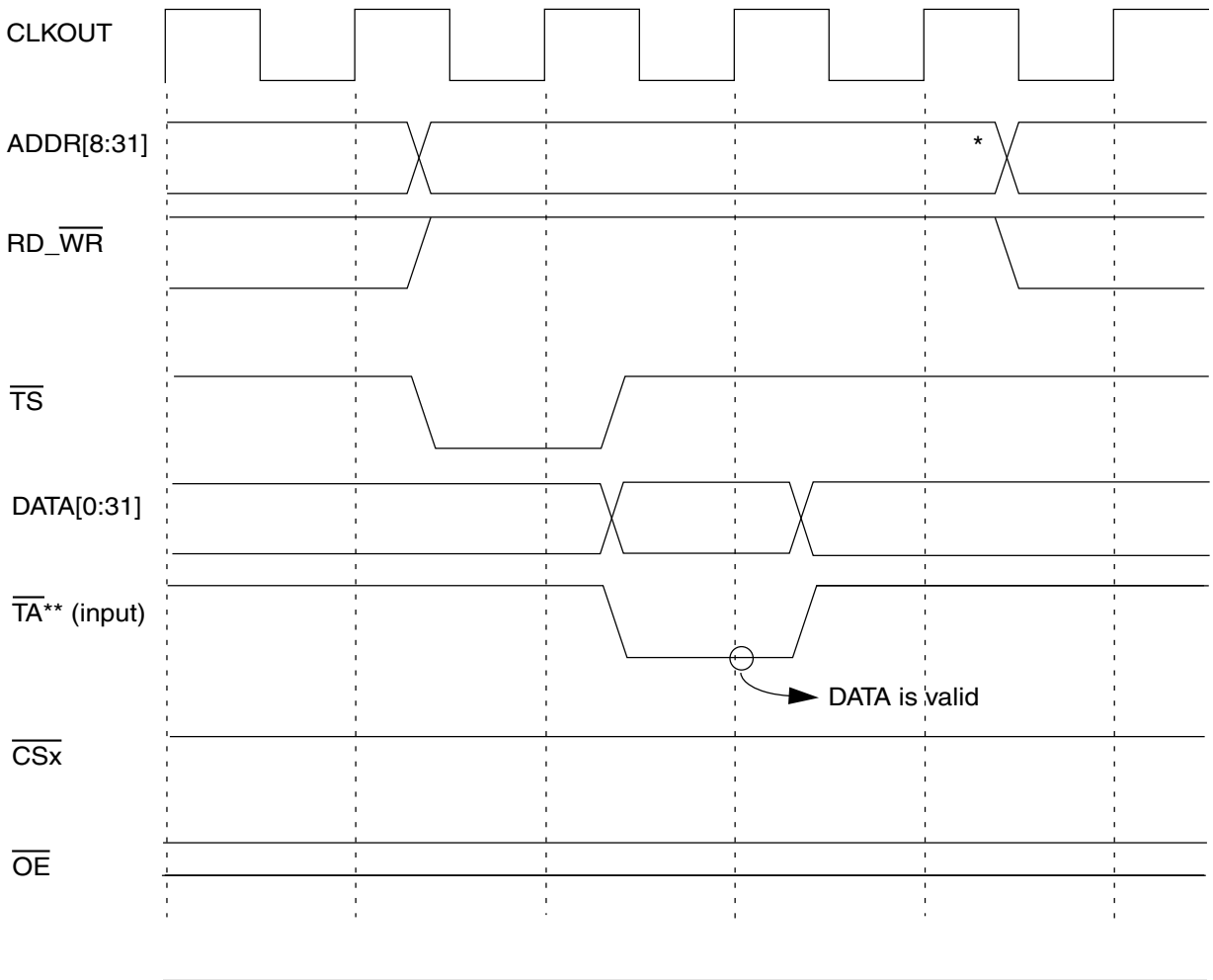
* There is no external TA signal. The TA signal is generated internally.

Figure 37-4. Single Beat 32-bit Read Cycle, CS Access, Zero Wait States



* There is no external TA signal. The TA signal is generated internally.

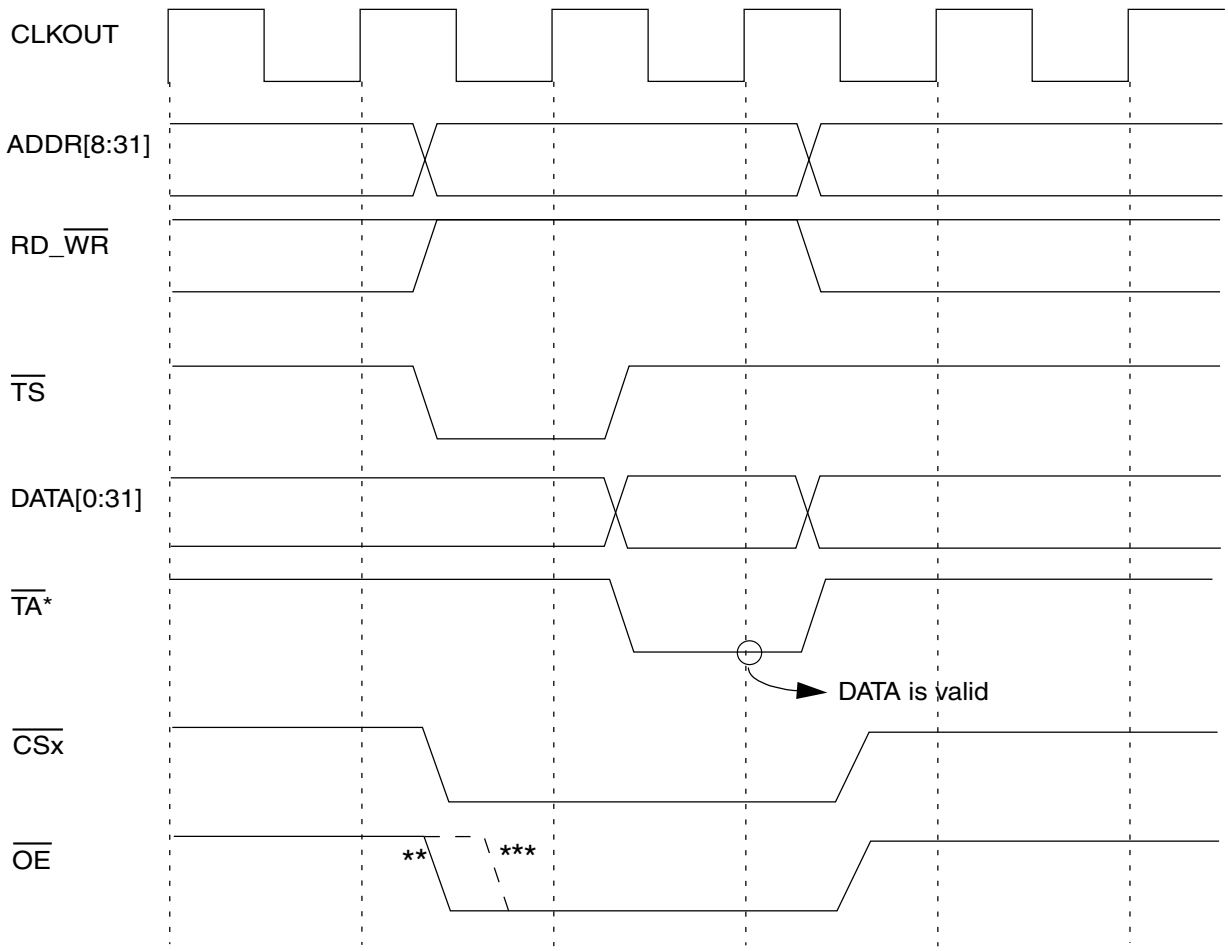
Figure 37-5. Single Beat 32-bit Read Cycle, CS Access, One Wait State



* The EBI drives address and control signals an extra cycle because it uses a latched version of the TA (1 cycle delayed) to terminate the cycle.

** There is no external TA signal. The TA signal is generated internally.

Figure 37-6. Single Beat 32-bit Read Cycle, Non-CS Access, Zero Wait States



* There is no external TA signal. The TA signal is generated internally.

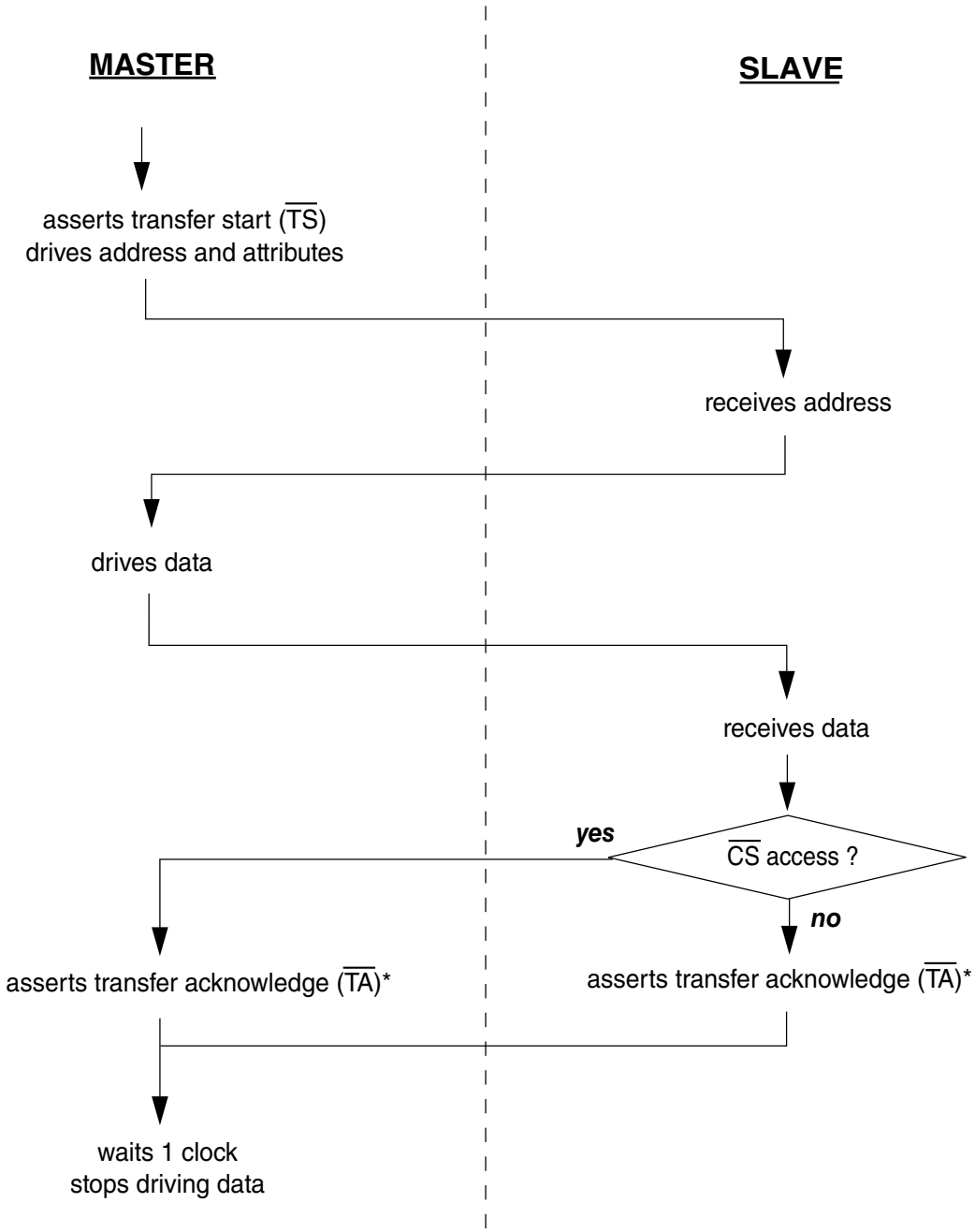
** When EBI_BR[EOE]=0b01 (or 0b10 in 1:1 bus speed mode), OE asserts around the same time as TS (could be slightly before or after TS, order of TS/OE assertion is not guaranteed).

*** When EBI_BR[EOE]=0b10 (and not in 1:1 bus speed mode), OE asserts 1 internal system clock (partial CLKOUT cycle) later as compared to TS.

Figure 37-7. Single Beat 32-bit Read Cycle, CS Access, Zero Wait States (EOE=0b01, 0b10)

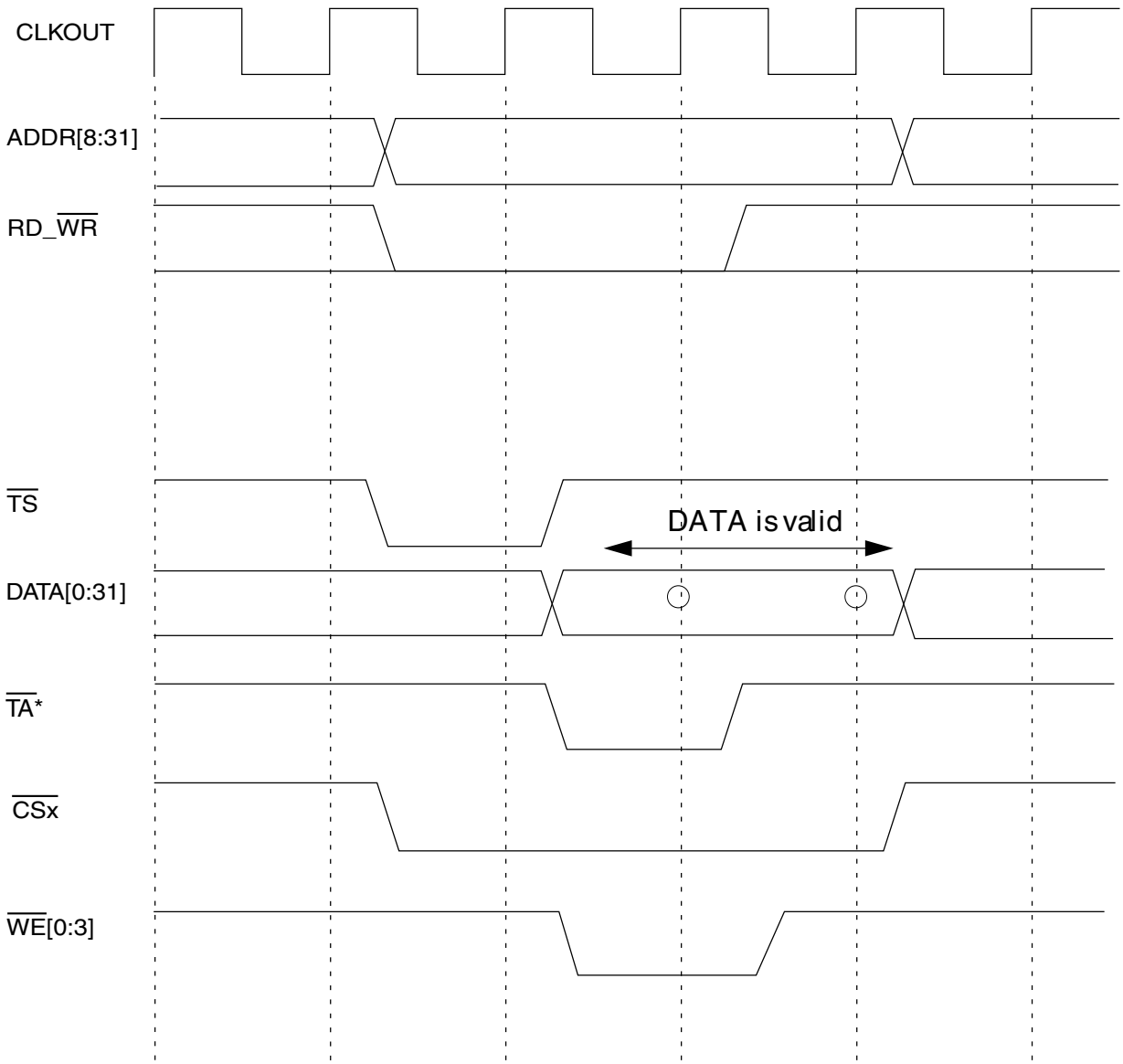
37.4.2.4.2 Single Beat Write Flow

The handshakes for a single beat write cycle are illustrated in the following flow and timing diagrams.



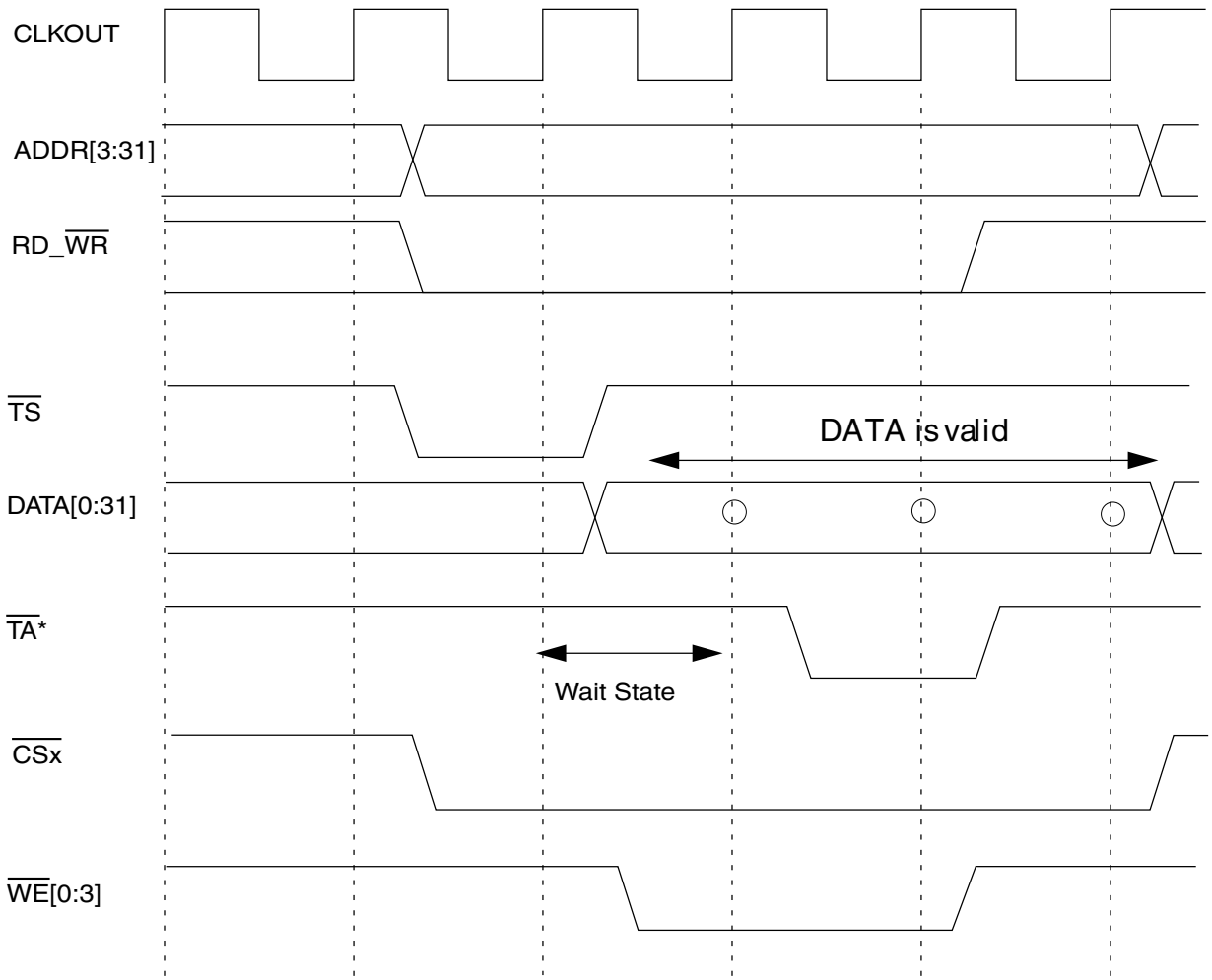
* There is no external TA signal. The TA signal is generated internally.

Figure 37-8. Basic Flow Diagram of a Single Beat Write Cycle



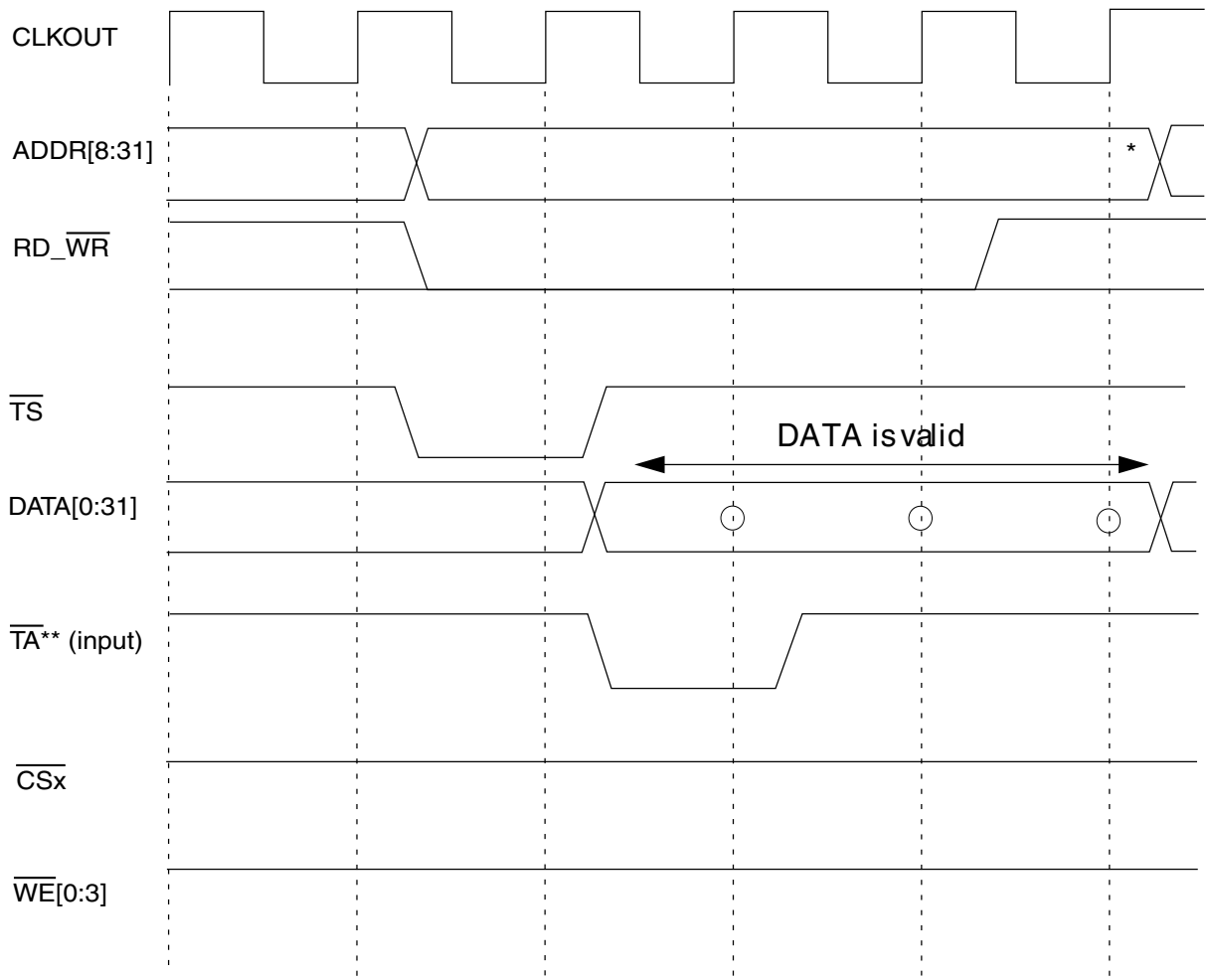
There is no external TA signal. The TA signal is generated internally.

Figure 37-9. Single Beat 32-bit Write Cycle, CS Access, Zero Wait States



* There is no external TA signal. The TA signal is generated internally.

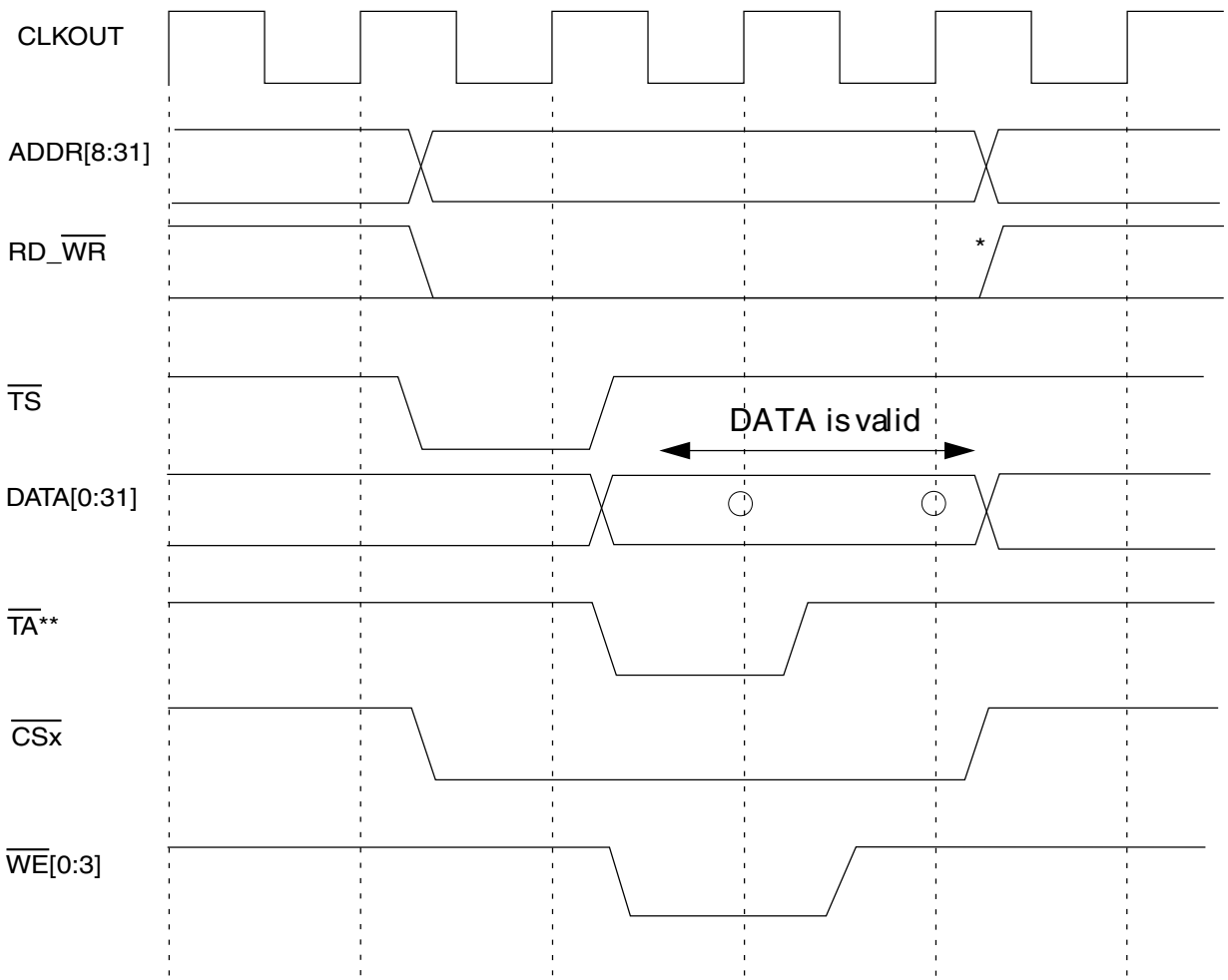
Figure 37-10. Single Beat 32-bit Write Cycle, CS Access, One Wait State



* The EBI drives address and control signals an extra cycle because it uses a latched version of the external TA (1 cycle delayed) to terminate the cycle.

** There is no external TA signal. The TA signal is generated internally.

Figure 37-11. Single Beat 32-bit Write Cycle, Non-CS Access, Zero Wait States



* Negation of RD_$\overline{WR}$ signal is delayed by 1 cycle (as shown) when EBI_BR[LWRN]=1.

** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-12. Single Beat 32-bit Write Cycle, $\overline{CS}$ Access, Zero Wait States, LWRN=1

37.4.2.4.3 Back-to-Back Accesses

Due to internal bus protocol, one dead cycle is necessary between back-to-back external bus accesses that are not part of a set of small accesses (see Small Accesses (Small Port Size and Short Burst Length) for small access timing). A dead cycle refers to a cycle between the $\overline{TA}$ of a previous transfer and the $\overline{TS}$ of the next transfer.

Note

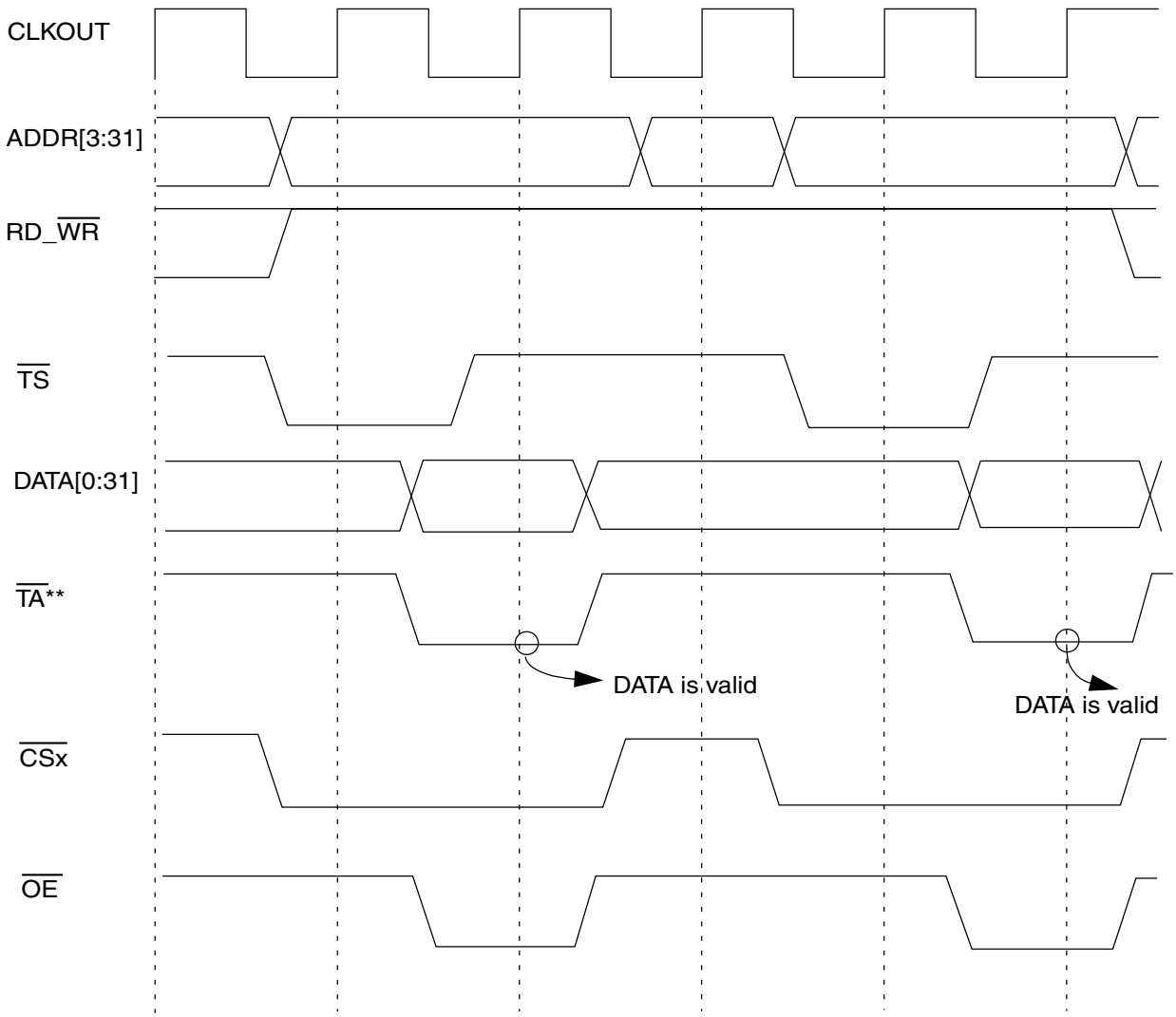
In some cases, $\overline{CS}$ remains asserted during this dead cycle, such as the cases of back-to-back writes or read-after-write to the same chip-select (with EBI_BR[GCSN]=0). See Figure 37-16 and Figure 37-18. However, if EBI_BR[GCSN]=1 for the first

access (see Figure 37-17 and Figure 37-19), then the EBI inserts an extra dead cycle between the accesses for these cases in order to guarantee $\overline{CS}$ negation between accesses.

Besides this dead cycle, in most cases, back-to-back accesses on the external bus do not cause any change in the timing from that shown in the previous diagrams, and the two transactions are independent of each other. The only exceptions to this are listed below:

- 4-word burst transfers (using 64-bit AMBA bus) whose starting address is not 0x20 aligned. In these cases, an extra cycle is required between the end of the 2nd 4-word burst access and the $\overline{TS}$ assertion of a subsequent access. See Figure 37-29.

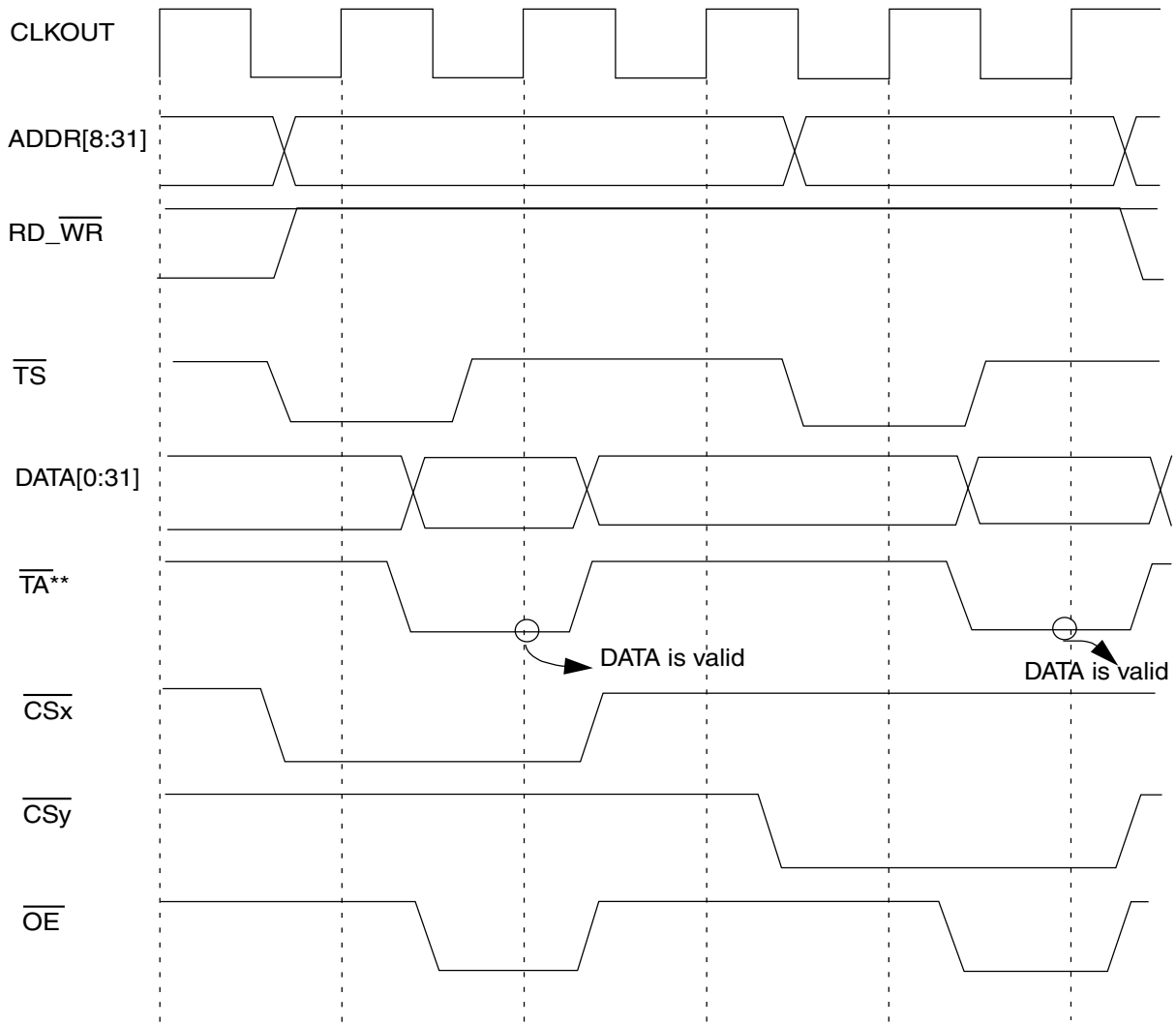
The following diagrams show a few examples of back-to-back accesses on the external bus.



* This timing diagram is unaffected by EBI_BR[GCSN].

** There is no external TA signal. The TA signal is generated internally.

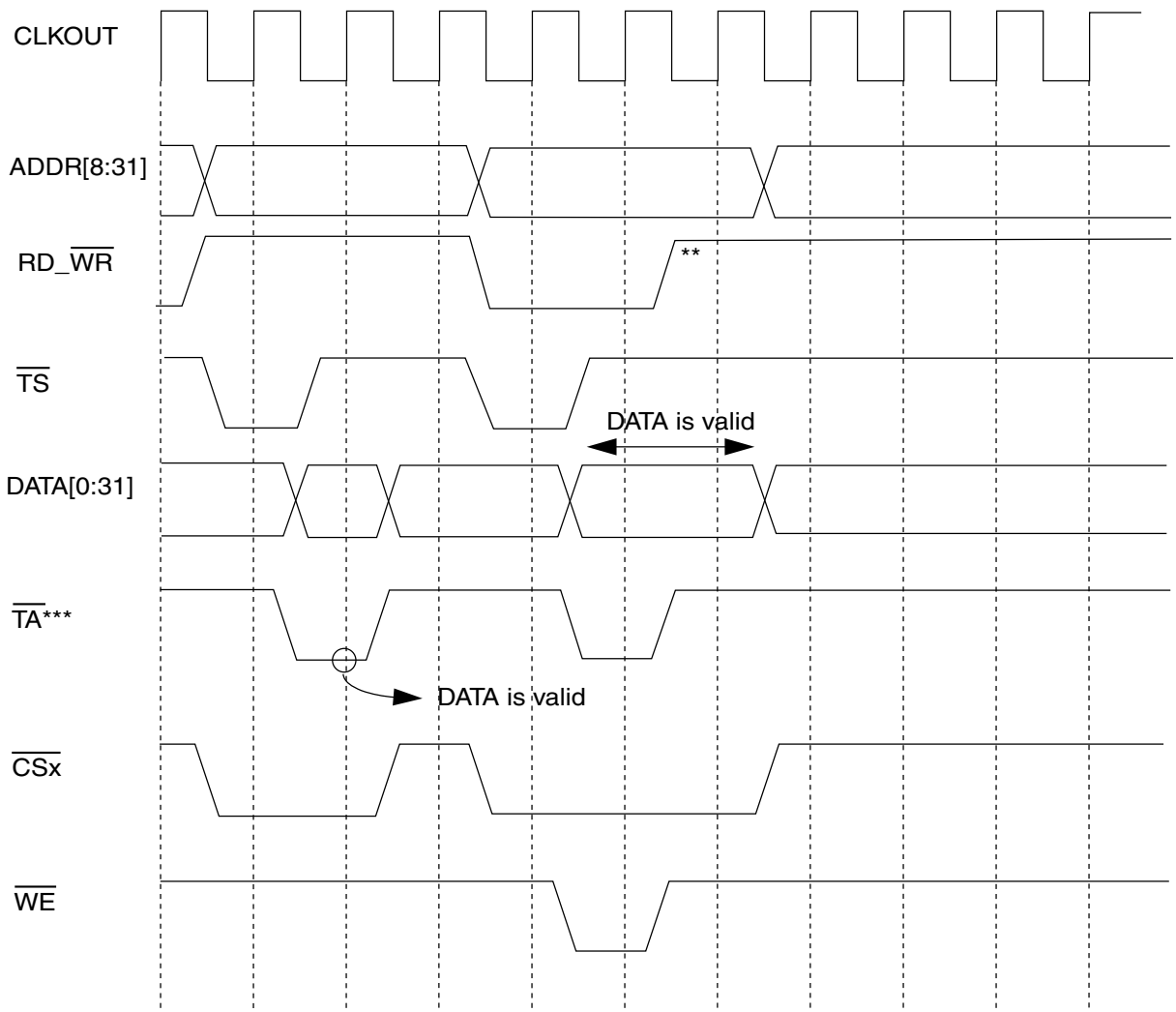
Figure 37-13. Back-to-Back 32-bit Reads to the Same CS Bank



* This timing diagram is unaffected by EBI_BR[GCSN].

** There is no external TA signal. The TA signal is generated internally.

Figure 37-14. Back-to-Back 32-bit Reads to Different CS Banks

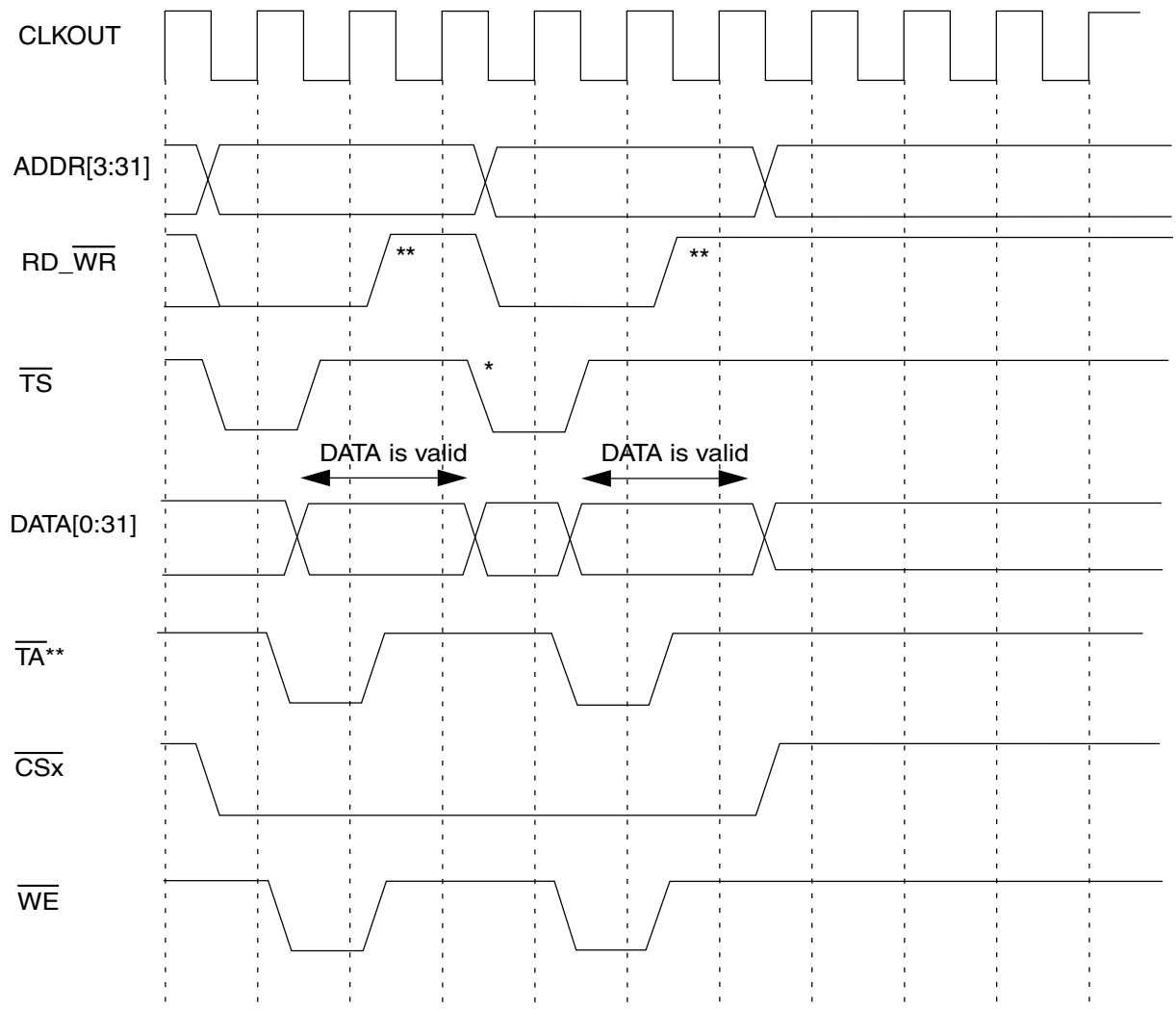


* This timing diagram is unaffected by EBI_BR[GCSN].

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_WR negation is delayed by 1 cycle.

*** There is no external TA signal. The TA signal is generated internally.

Figure 37-15. Write After Read to the Same CS Bank

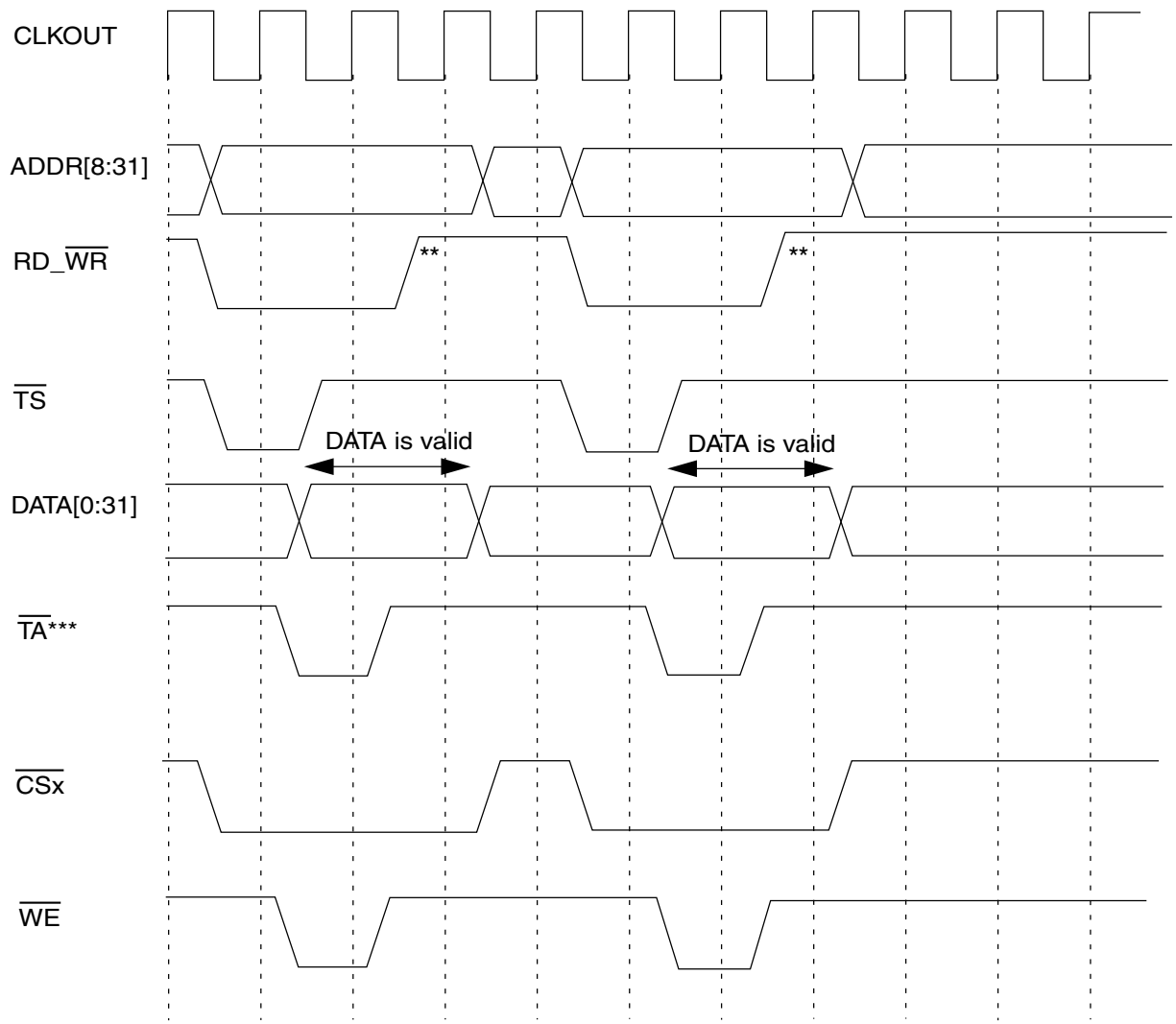


* Timing shown applies when EBI_BR[GCSN]=0. When EBI_BR[GCSN]=1, the 2nd $\overline{TS}$ assertion is delayed by 1 cycle and $\overline{CS}$ negates between the two transfers.

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle, such that it does not negate between these two b-t-b transfers.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-16. Back-to-Back 32-bit Writes to the Same CS Bank, GCSN=0

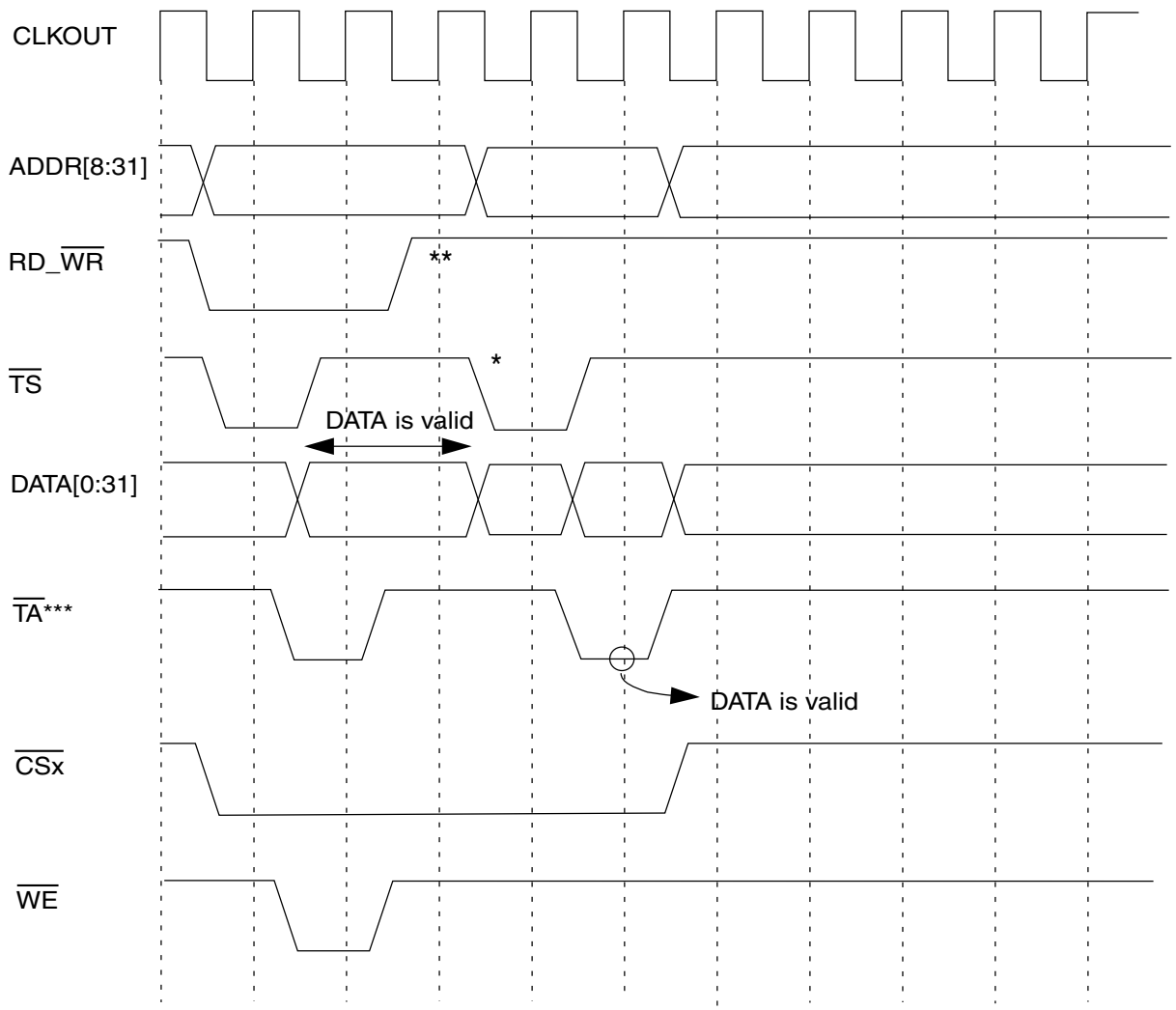


* Timing shown applies when EBI_BR[GCSN]=1.

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-17. Back-to-Back 32-bit Writes to the Same $\overline{CS}$ Bank, GCSN=1a

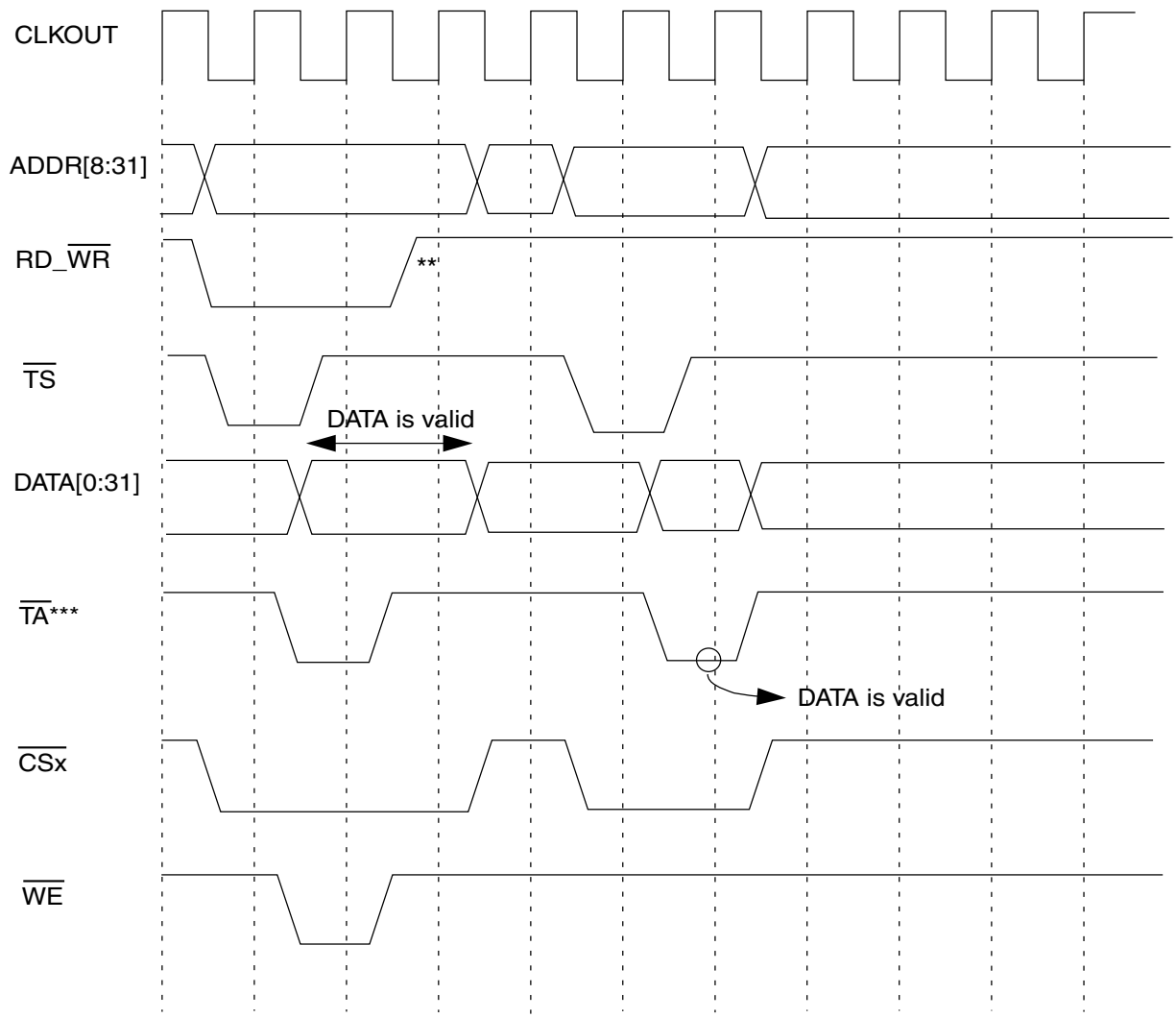


* Timing shown applies when EBI_BR[GCSN]=0. When EBI_BR[GCSN]=1, the 2nd $\overline{TS}$ assertion is delayed by 1 cycle and $\overline{CS}$ negates between the two transfers.

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-18. Read After Write to the Same $\overline{CS}$ Bank, GCSN=0

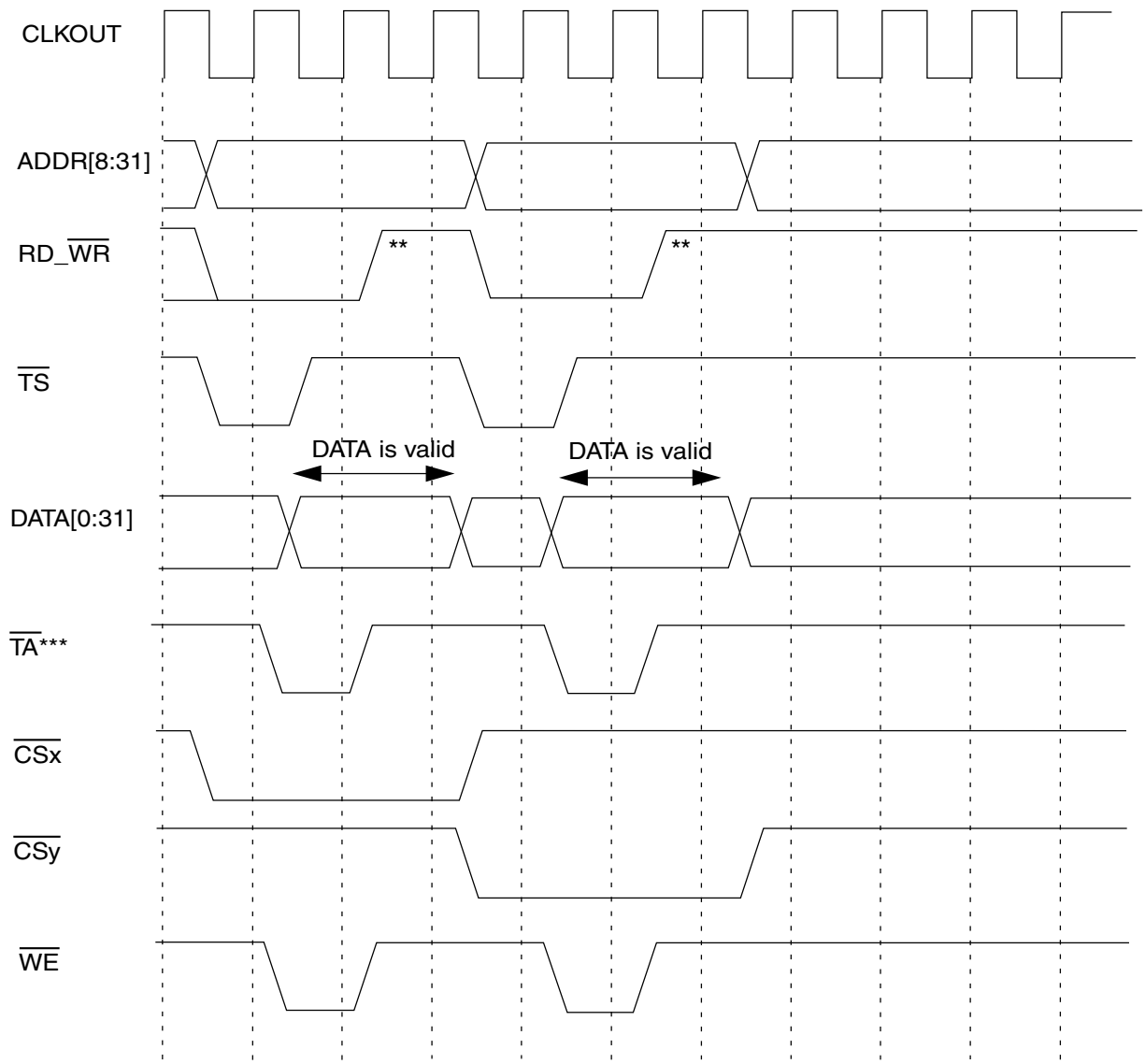


* Timing shown applies when EBI_BR[GCSN]=1.

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-19. Read After Write to the Same $\overline{CS}$ Bank, GCSN=1

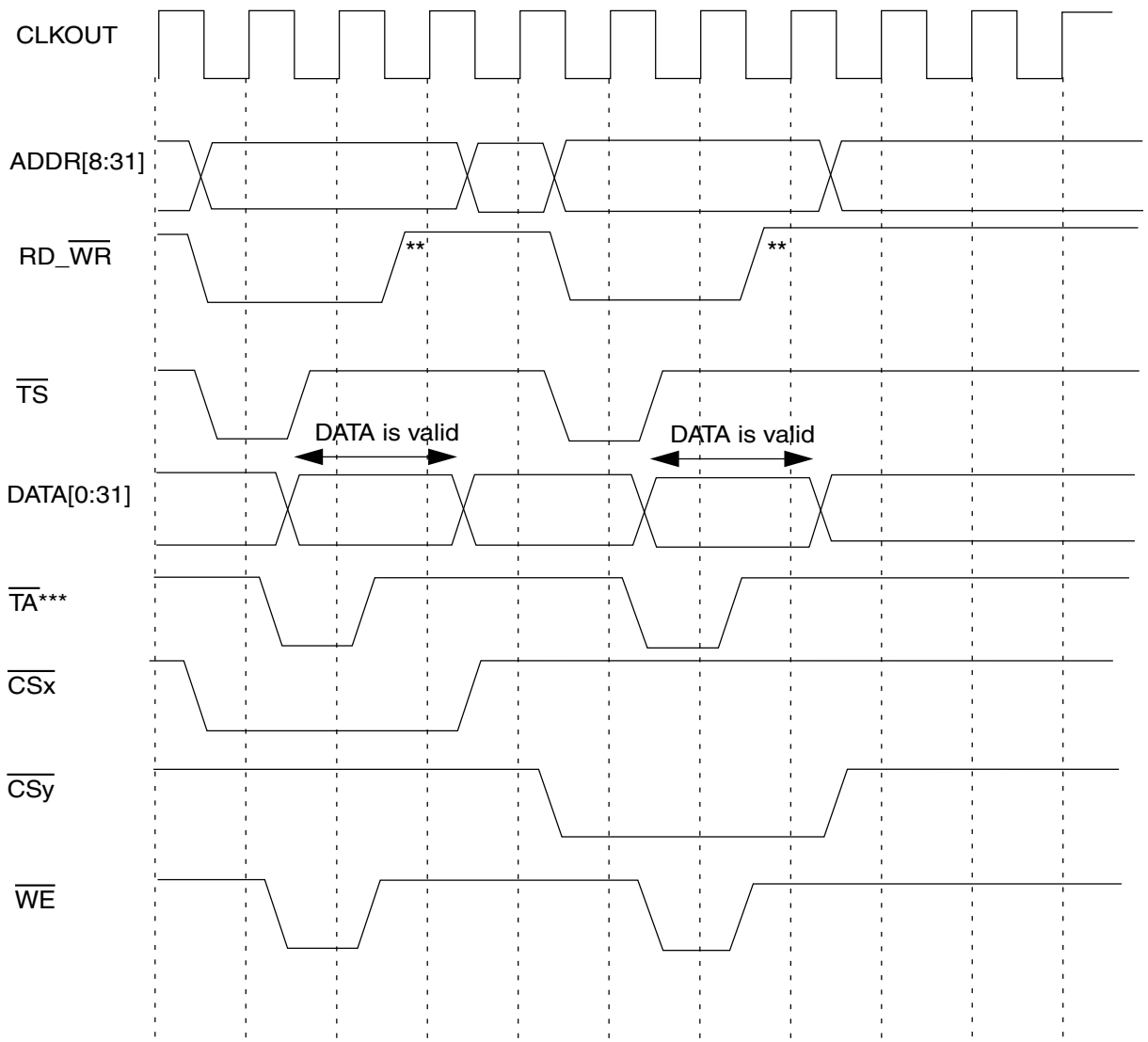


* This timing diagram applies when EBI_BR[GCSN]=0.

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle, such that it does not negate between these two b-t-b transfers.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-20. Back-to-Back 32-bit Writes to Different $\overline{CS}$ Banks (GCSN=0)

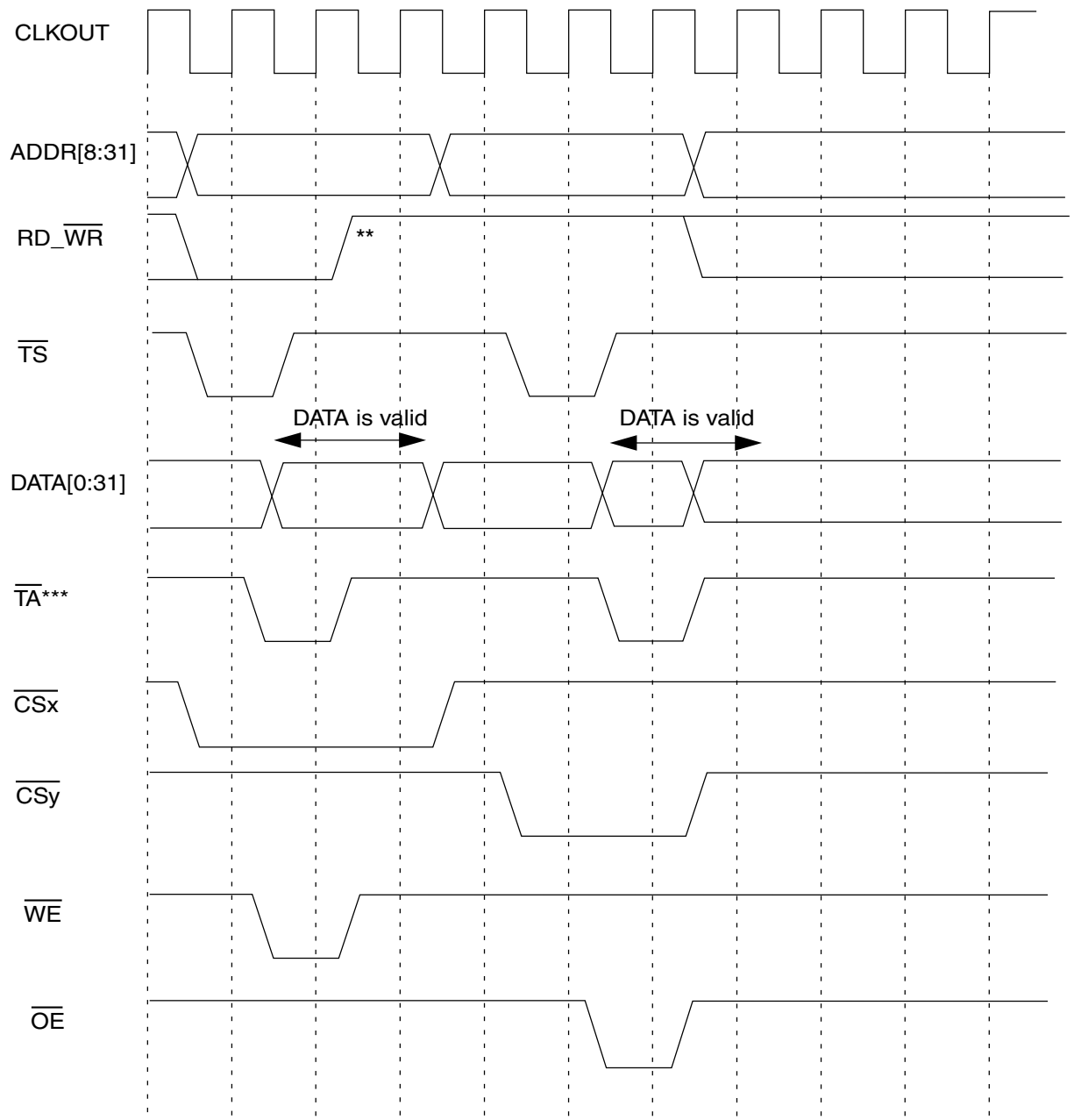


* Timing shown applies when EBI_BR[GCSN]=1.

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-21. Back-to-Back 32-bit Writes to the Same $\overline{CS}$ Bank, GCSN=1b



* This timing diagram is unaffected by EBI_BR[GCSN].

** Timing shown applies when EBI_BR[LWRN]=0. When EBI_BR[LWRN]=1, RD_$\overline{WR}$ negation is delayed by 1 cycle.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-22. Read After Write to Different $\overline{CS}$ Banks

37.4.2.5 Burst Transfer

The EBI supports wrapping 32-byte critical-doubleword-first burst transfers. Bursting is supported only for internally-requested (e.g. core, dma, etc.) cache-line size (32-byte) read accesses to external devices that use the chip selects.

Note

In the case where the internal AMBA data bus width is 32 bits (instead of 64), the EBI supports a 16-byte critical-word-first burst transfer (instead of 32-byte critical-doubleword-first). In general, this document assumes the default case of 64-bit AMBA bus, 32-byte burst transfer.

Accesses to devices operating without a chip select are always single beat. If an internal request to the EBI indicates a size of less than 32 bytes, the request is fulfilled by running one or more single-beat external transfers, not by an external burst transfer.

An 8-word wrapping burst reads eight 32-bit words by supplying a starting address that points to one of the words (doubleword aligned) and requiring the memory device to sequentially drive each word on the data bus. The selected slave device must internally increment ADDR[27:29] (also ADDR30 in the case of a 16-bit port size device) of the supplied address for each transfer, until the address reaches an 8-word boundary, and then wrap the address to the beginning of the 8-word boundary. The address and transfer attributes supplied by the EBI remain stable during the transfers. Termination of each beat transfer occurs by the EBI asserting $\overline{\text{TA}}$. The EBI requires that addresses be aligned to a doubleword boundary on all burst cycles.

Table 37-4 shows the burst order of beats returned for an 8-word burst to a 32-bit port.

Table 37-4. Wrap Bursts Order

Burst Starting Address ADDR[27:28]	Burst Order (Assuming 32-bit Port Size)
00	word0 -> word1 -> word2 -> word3 -> word4 -> word5 -> word6 -> word7
01	word2 -> word3 -> word4 -> word5 -> word6 -> word7 -> word0 -> word1
10	word4 -> word5 -> word6 -> word7 -> word0 -> word1 -> word2 -> word3
11	word6 -> word7 -> word0 -> word1 -> word2 -> word3 -> word4 -> word5

The general case of burst transfers assumes that the external memory has 32-bit port size and 8-word burst length (SBL=0, BL=0). The EBI can also burst from 16-bit port size memories, taking twice as many external beats to fetch the data as compared to a 32-bit port with the same burst length. The EBI can also burst from 16-bit or 32-bit memories that have a 4-word burst length (SBL=0, BL=1 in the appropriate Base Register). In this case, two external 4-word burst transfers (wrapping on 4-word boundary) are performed

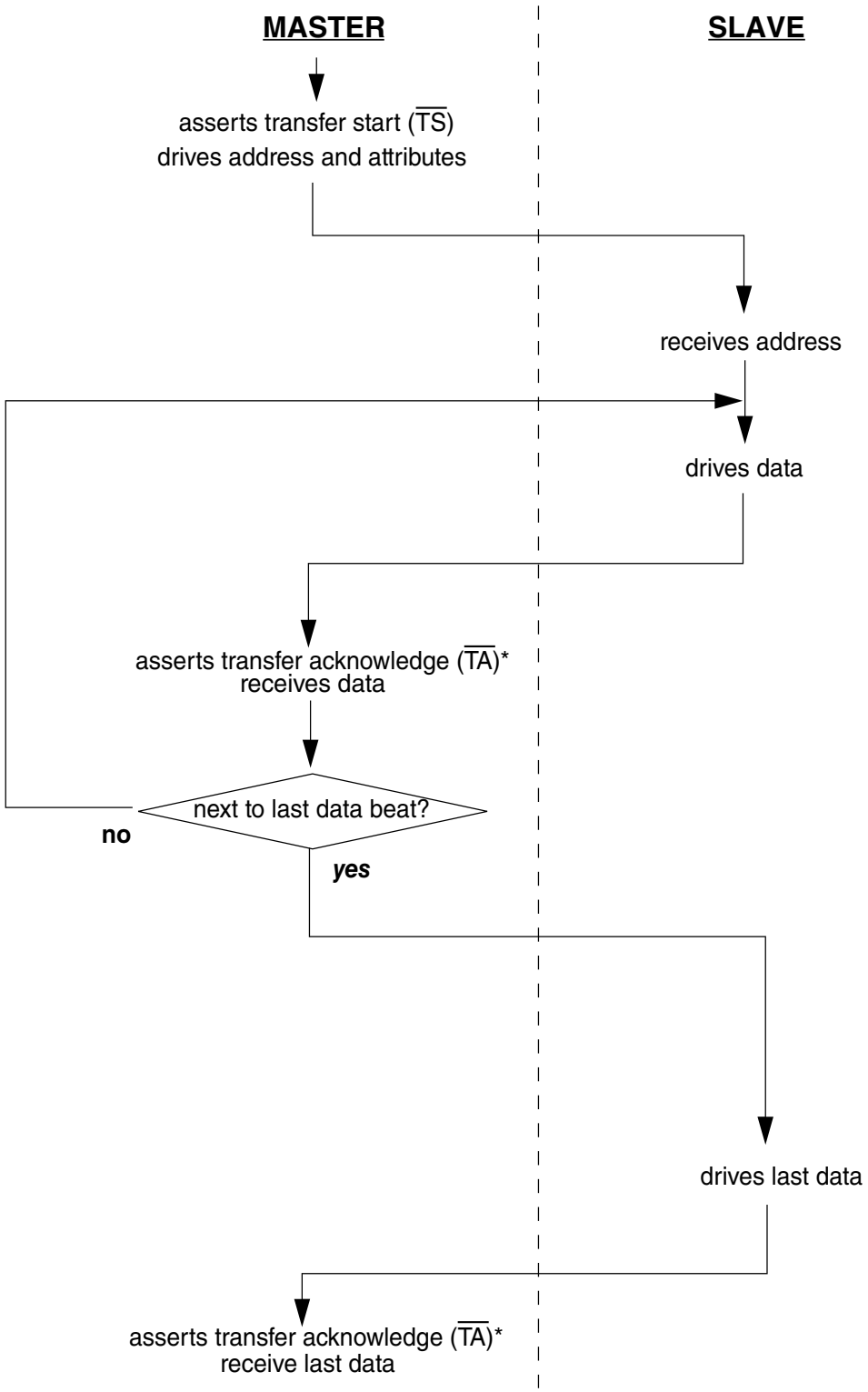
to fulfill the internal 8-word request[1]. This operation is considered atomic by the EBI, so the EBI does not allow other unrelated master accesses or bus arbitration to intervene between the transfers. For more details and a timing diagram, see Small Access Example #3: 32-byte Read to 32-bit Port with BL=1.

The EBI can also burst from 16-bit or 32-bit memories that have a 2-word burst length (SBL=1, BL=X in the appropriate Base Register). In this case, four[2] external 2-word burst transfers (wrapping on 2-word boundary) are performed to fulfill the internal 8-word request. This operation is considered atomic by the EBI, so the EBI does not allow other unrelated master accesses or bus arbitration to intervene between the transfers. For more details and a timing diagram, see Small Access Example #5: 32-byte Read to 32-bit Port with SBL=1.

During burst cycles, the $\overline{\text{BDIP}}$ (Burst Data In Progress) signal is used to indicate the duration of the burst data. During the data phase of a burst read cycle, the EBI receives data from the addressed slave. If the EBI needs more than one data, it asserts the $\overline{\text{BDIP}}$ signal. Upon receiving the data prior to the last data, the EBI negates $\overline{\text{BDIP}}$. Thus, the slave stops driving new data after it receives the negation of $\overline{\text{BDIP}}$ on the rising edge of the clock. Some slave devices have their burst length and timing configurable internally and thus may not support connecting to a BDIP pin. In this case, $\overline{\text{BDIP}}$ is driven by the EBI normally, but the output is ignored by the memory and the burst data behavior is determined by the internal configuration of the EBI and slave device. When the TBDIP bit is set in the appropriate Base Register, the timing for $\overline{\text{BDIP}}$ is altered. See TBDIP Effect on Burst Transfer for this timing.

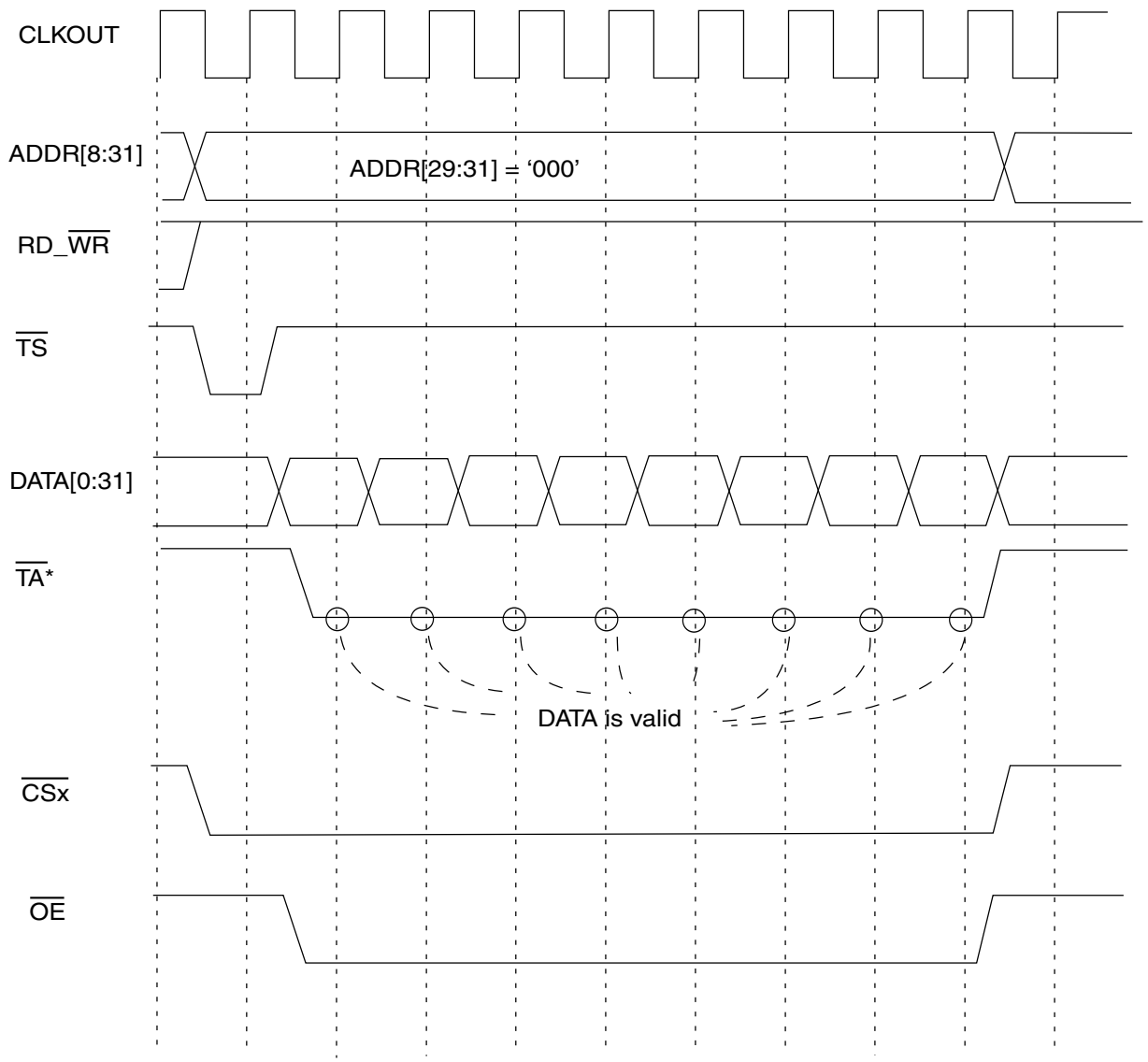
Since burst writes are not supported by the EBI, the EBI negates $\overline{\text{BDIP}}$ during write cycles.

1. This case (of 2 external 4-word burst transfers being required) applies only to AMBA data bus width of 64 bits.
2. Applies to 64-bit AMBA data bus width. For 32-bit AMBA data bus width, only two transfers are performed.



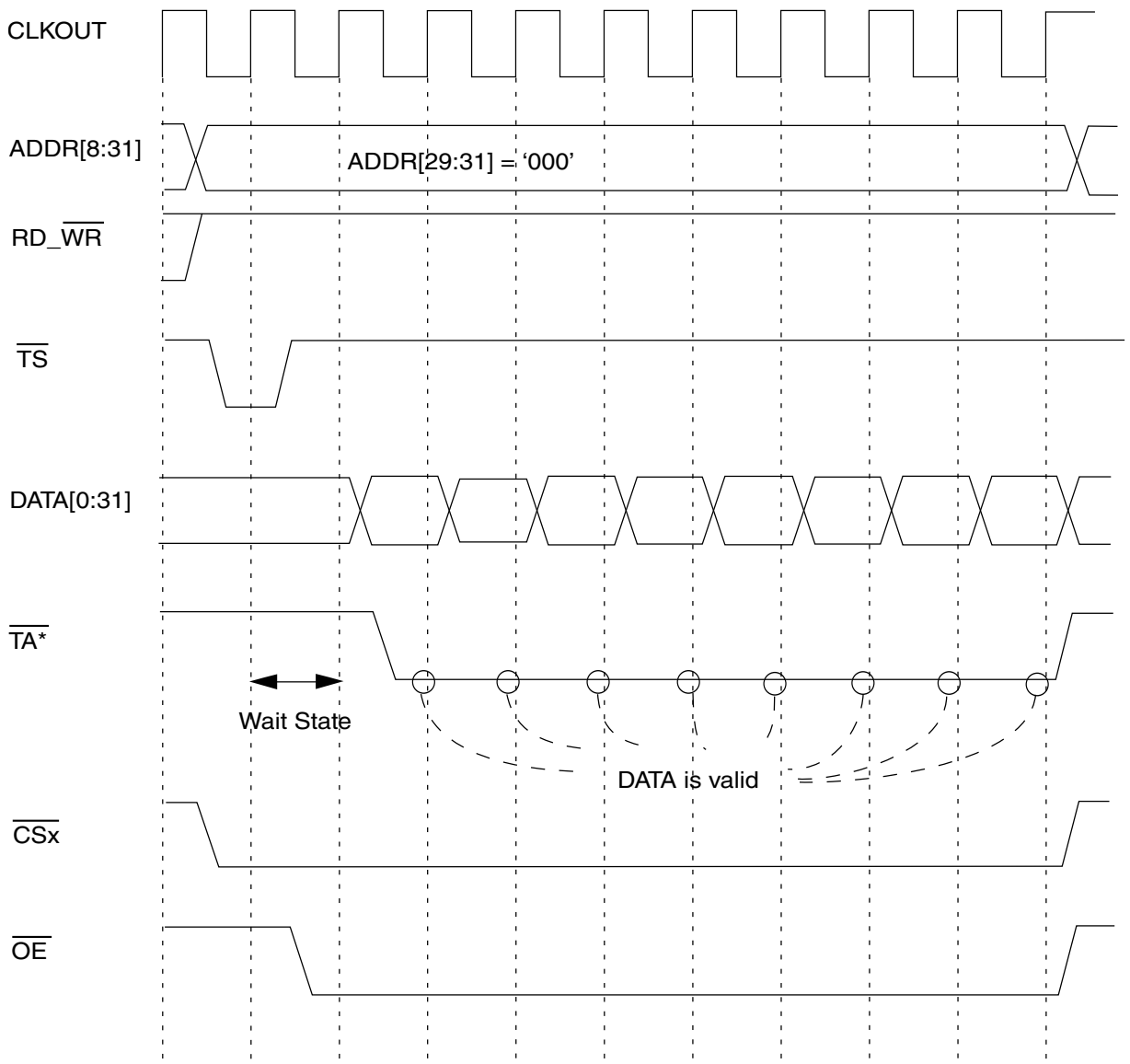
*There is no external TA signal. The TA signal is generated internally.

Figure 37-23. Basic Flow Diagram of a Burst Read Cycle



* There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-24. Burst 32-bit Read Cycle, Zero Wait States

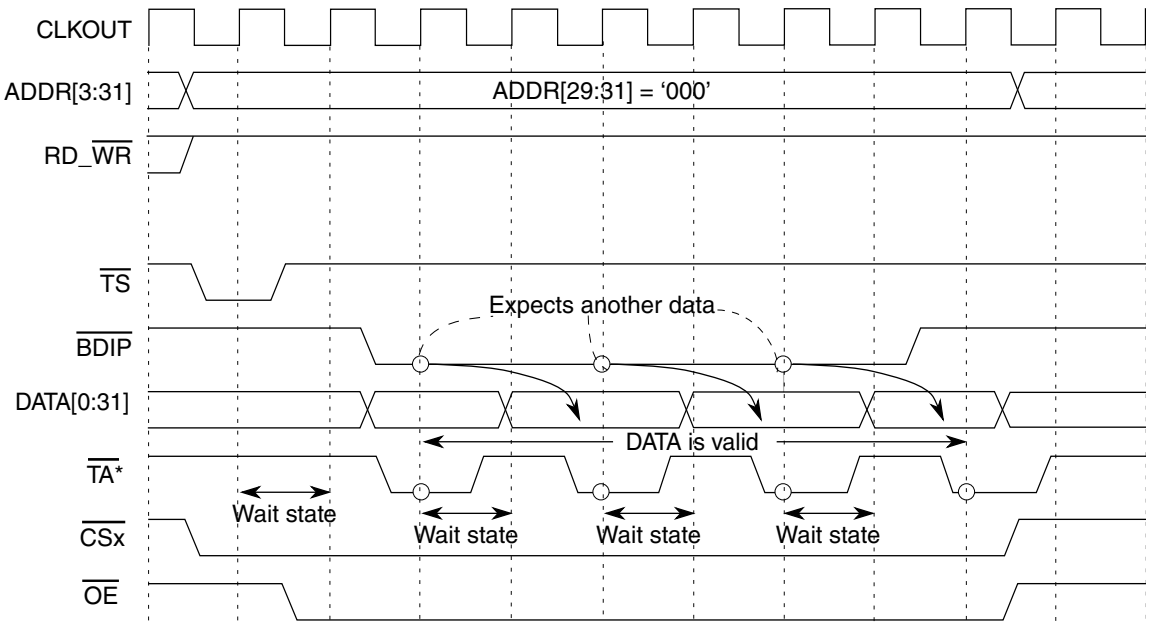


* There is no external TA signal. The TA signal is generated internally.

Figure 37-25. Burst 32-bit Read Cycle, One Initial Wait State

37.4.2.5.1 TBDIP Effect on Burst Transfer

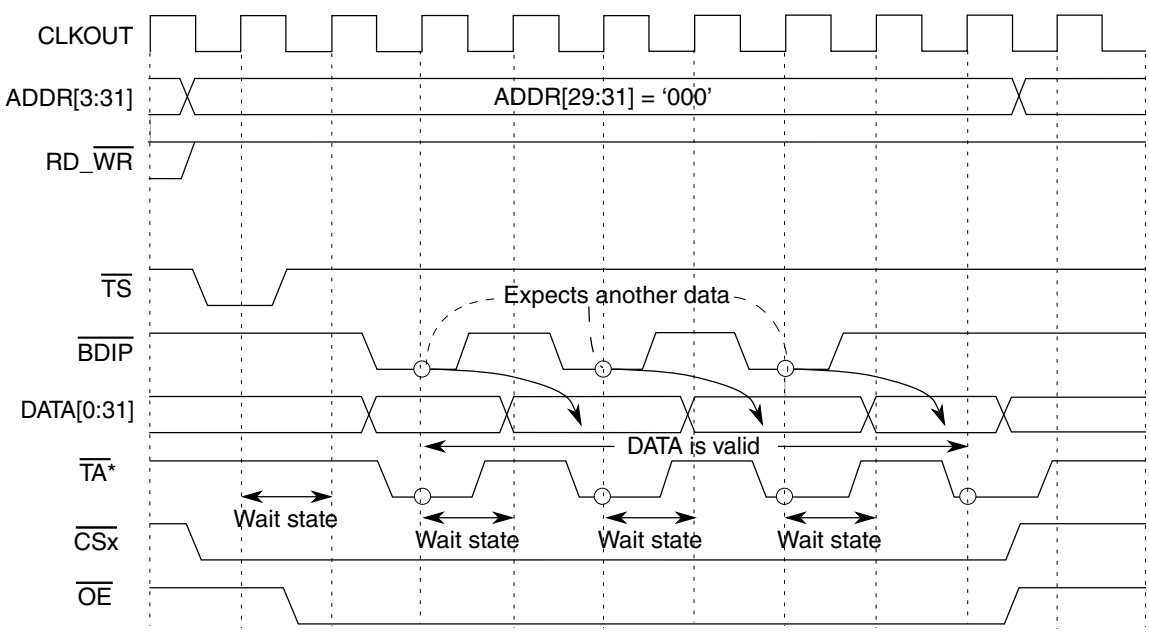
Some memories require different timing on the BDIP signal than the default to run burst cycles. Using the default value of TBDIP=0 in the appropriate EBI Base Register results in BDIP being asserted (SCY+1) cycles after the address transfer phase, and being held asserted throughout the cycle regardless of the wait states between beats (BSCY). Figure 37-26 shows an example of the TBDIP=0 timing for a 4-beat burst with BSCY=1.



*There is no external TA signal. The TA signal is generated internally.

Figure 37-26. Burst 32-bit Read Cycle, One Wait State between Beats, TBDIP=0

When using TBDIP=1, the BDIP behavior changes to toggle between every beat when BSCY is a non-zero value. Figure 37-27 shows an example of the TBDIP=1 timing for the same 4-beat burst shown in Figure 37-26.



*There is no external TA signal. The TA signal is generated internally.

Figure 37-27. Burst 32-bit Read Cycle, One Wait State between Beats, TBDIP=1

37.4.2.6 Small Accesses (Small Port Size and Short Burst Length)

In this context, a *small access* refers to an access whose burst length and port size (SBL, BL, PS bits in Base Register for chip-select access or default burst disabled, 32-bit port for non-chip-select access) are such that the number of bytes requested by the internal master cannot all be fetched (or written) in one external transaction. If this is the case, the EBI initiates multiple transactions until all the requested data is transferred. It should be noted that all the transactions initiated to complete the data transfer are considered as an atomic transaction, so the EBI does not allow other unrelated master accesses to intervene between the transfers.

Table 37-5 shows all the combinations of burst length, port size, and requested byte count that cause the EBI to run multiple external transactions to fulfill the request.

Table 37-5. Small Access Cases

Byte Count Requested by internal master	Burst Length	Port Size	# External Accesses to Fulfill Request
Non-Burstable Chip-Select Banks (BI=1)			
4	1 beat	16-bit	2
8	1 beat	32-bit	2
8	1 beat	16-bit	4
16[1]	1 beat	32-bit	4
16[1]	1 beat	16-bit	8
32[2]	1 beat	32-bit	8
32[2]	1 beat	16-bit	16
Burstable Chip-Select Banks (BI=0)			
32[2]	4 words	16-bit (8 beats), 32-bit (4 beats)	2
32[3]	2 words	16-bit (4 beats), 32-bit (2 beats)	4

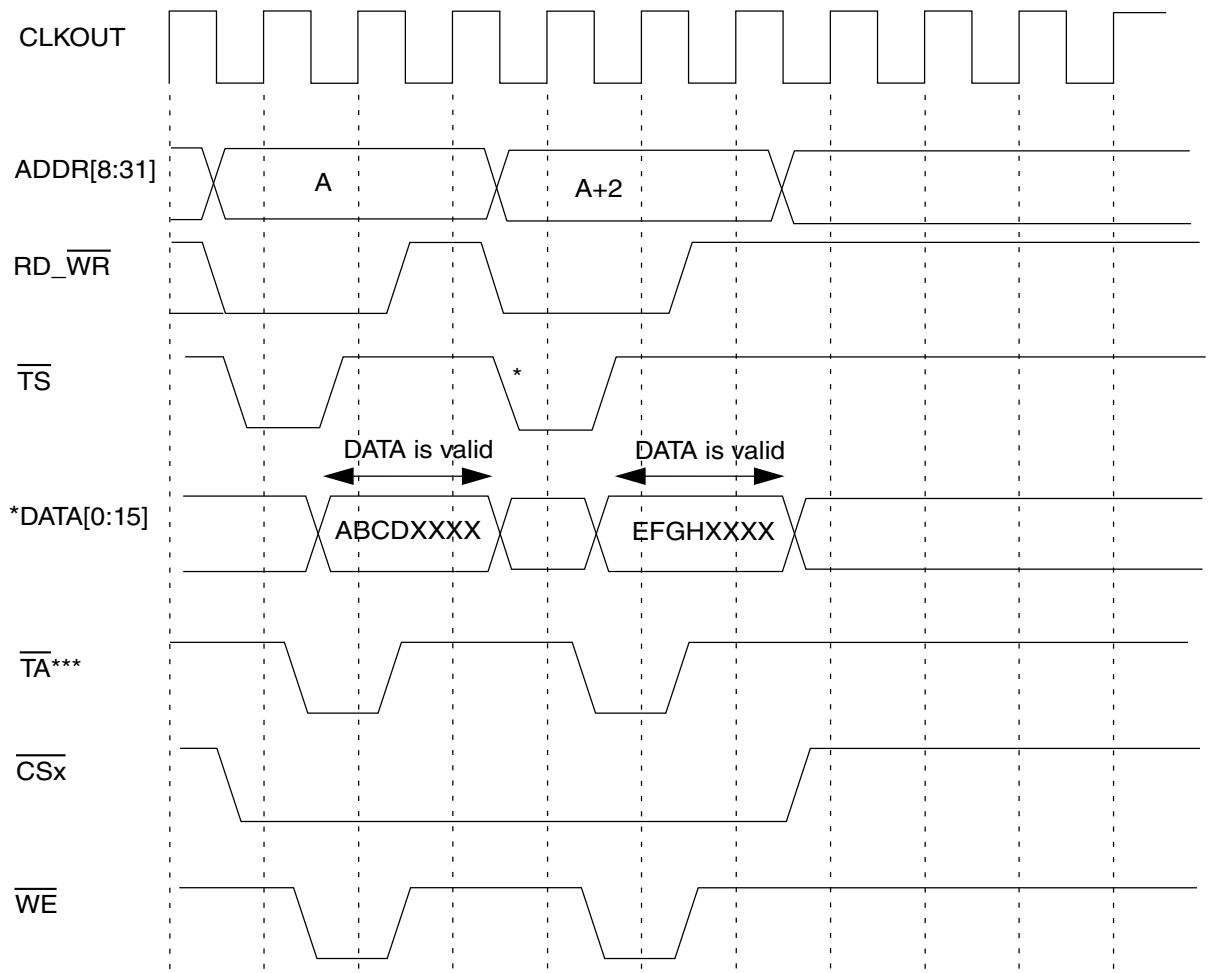
1. Only supported for case of 32-bit internal AMBA data bus.
2. Only supported for case of 64-bit internal AMBA data bus.
3. SBL=1 (2-word burst case).

In most cases, the timing for small accesses is the same as for normal single-beat and burst accesses, except that multiple back-to-back external transfers are executed for each internal request. These transfers have no additional dead cycles in-between that are not present for back-to-back stand-alone transfers except for the case of writes with an internal request size of >64 bits.

The following sections show a few examples of small accesses. The timing for the remaining cases in Table 37-5 can be extrapolated from these and the other timing diagrams in this document.

37.4.2.6.1 Small Access Example #1: 32-bit Write to 16-bit Port

Figure 37-28 shows an example of a 32-bit write to a 16-bit port, requiring two 16-bit external transactions.



* Or DATA[16:31], based on D16_31 bit in EBI_MCR.

** Timing shown applies when EBI_BR[GCSN]=0. When EBI_BR[GCSN]=1, the 2nd TS assertion is delayed by 1 cycle and CS negates between the two transfers..

*** There is no external TA signal. The TA signal is generated internally.

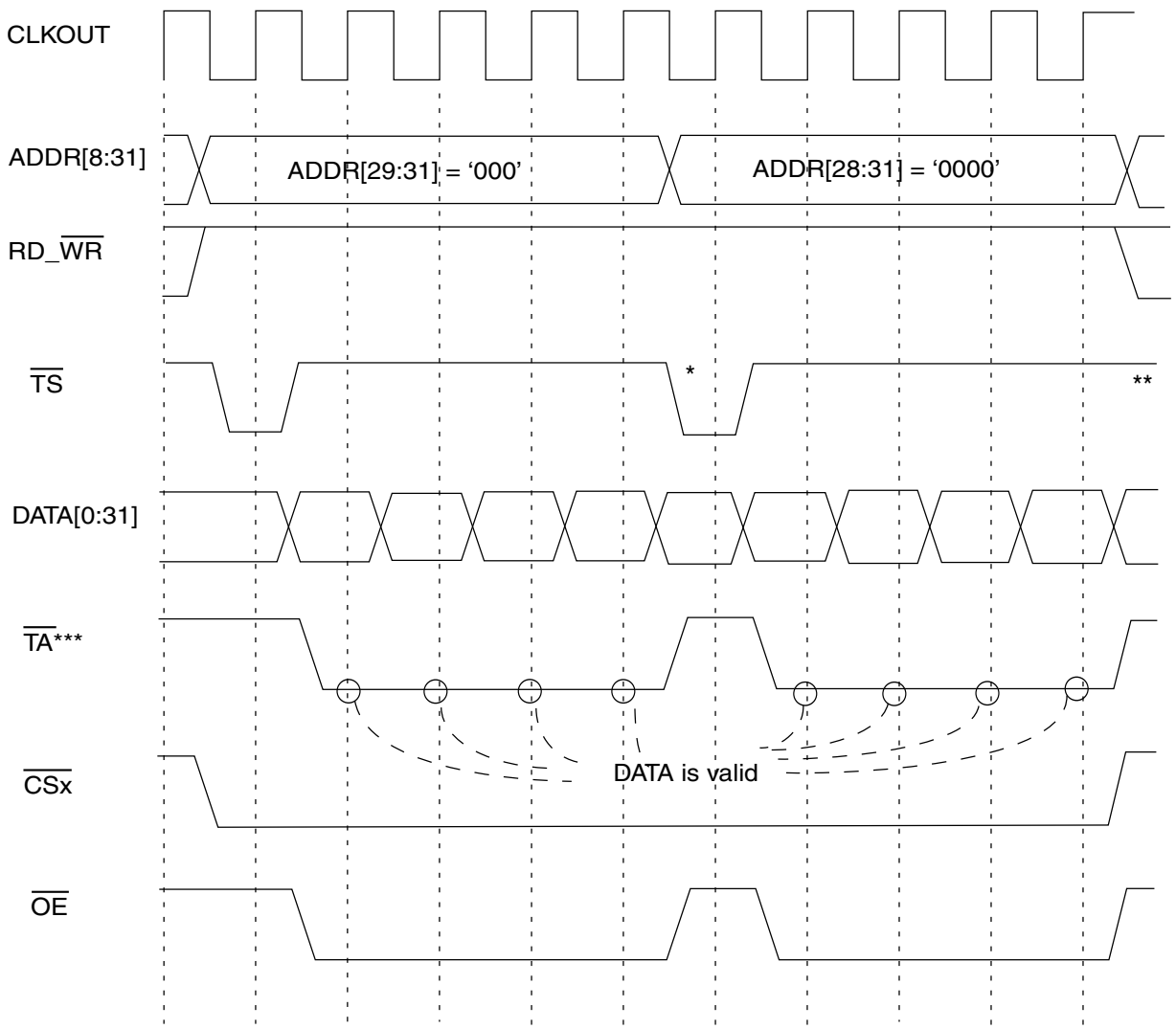
Figure 37-28. Single Beat 32-bit Write Cycle, 16-bit Port Size, Basic Timing

37.4.2.6.2 Small Access Example #3: 32-byte Read to 32-bit Port with BL=1

Figure 37-29 shows an example of a 32-byte read to a 32-bit burst enabled port with burst length of 4 words, requiring two 4-word (16-byte) external transactions. For this case, the address for the 2nd 4-word burst access is calculated by adding 0x10 to the lower 5 bits of the 1st address (no carry), and then masking out the lower 4 bits to fix them at zero.

Table 37-6. Examples of 4-word Burst Addresses

1st Address	Lower 5 bits of 1st Address + 0x10 (no carry)	Final 2nd Address (After Masking Lower 4 Bits)
0x000	0x10	0x10
0x008	0x18	0x10
0x010	0x00	0x00
0x018	0x08	0x00
0x020	0x30	0x30
0x028	0x38	0x30
0x030	0x20	0x20
0x038	0x28	0x20



* Timing shown applies when EBI_BR[GCSN]=0. When EBI_BR[GCSN]=1, the 2nd TS assertion is delayed by 1 cycle and CS negates between the two transfers.

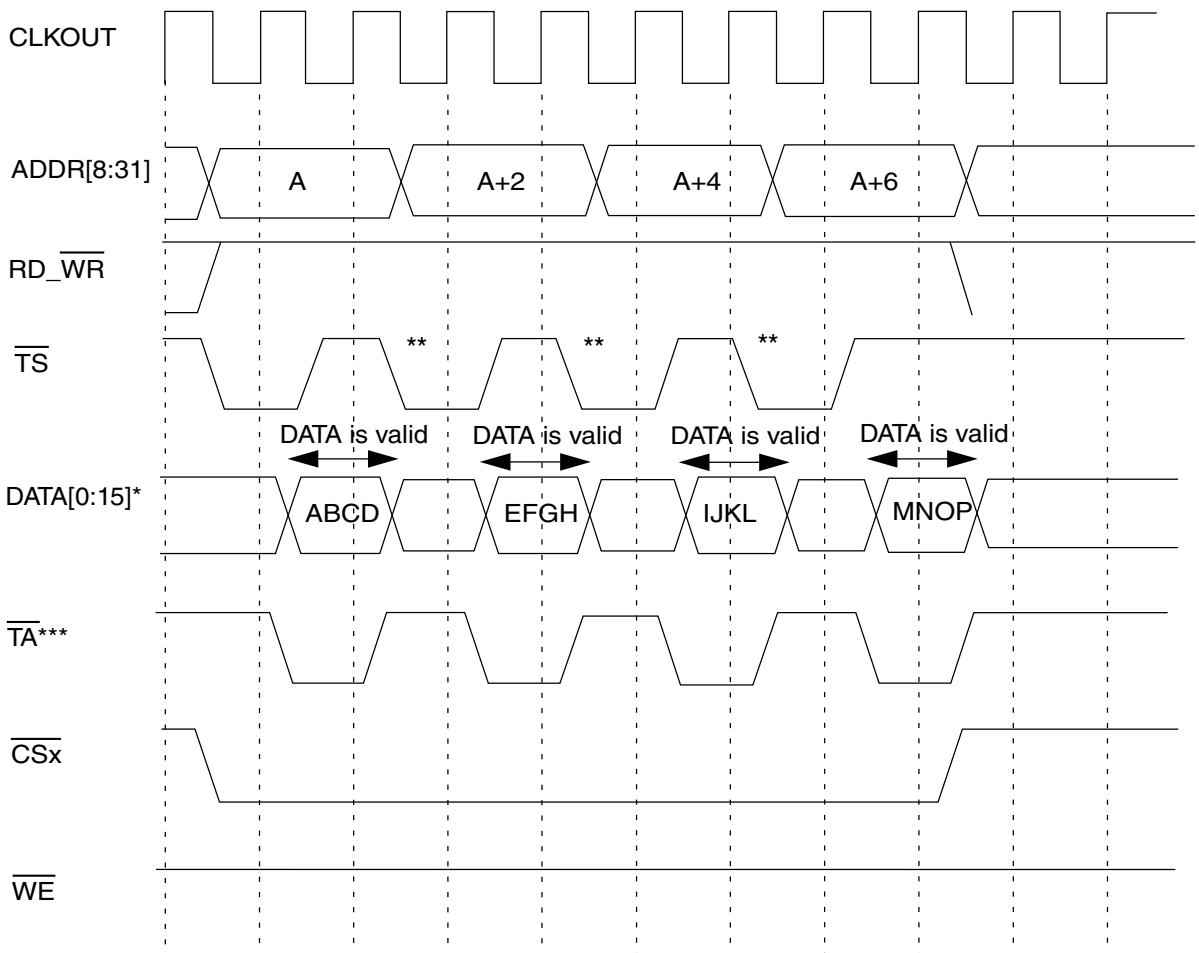
** For the case where the starting address for 4-word burst is not 0x20 aligned, the start of a subsequent back-to-back transfer is delayed by 1 cycle.

*** There is no external TA signal. The TA signal is generated internally.

Figure 37-29. 32-Byte Read with B-T-B 4-Word Bursts to 32-bit Port, Zero Wait States

37.4.2.6.3 Small Access Example #4: 64-bit Read to 16-bit Port

Figure 37-30 shows an example of a 64-bit read to a 16-bit port, requiring four 16-bit external transactions.



* Or DATA[16:31], based on D16_31 bit in EBI_MCR.

** Timing shown applies when EBI_BR[GCSN]=0. When EBI_BR[GCSN]=1, the 2nd-4th $\overline{TS}$ assertions are each delayed by 1 cycle and $\overline{CS}$ negates between each transfer.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

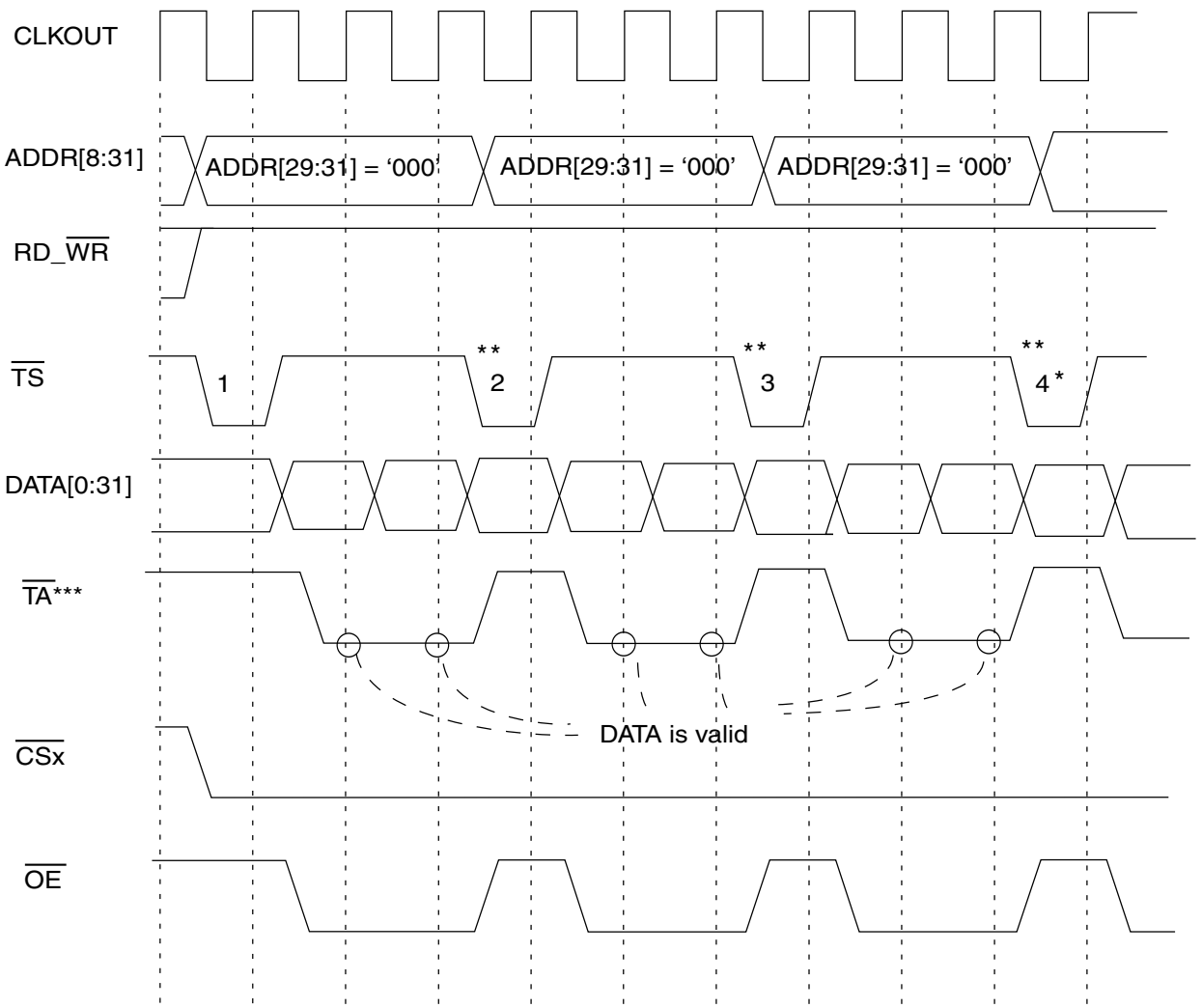
Figure 37-30. Single Beat 64-bit Read Cycle, 16-bit Port Size, Basic Timing

37.4.2.6.4 Small Access Example #5: 32-byte Read to 32-bit Port with SBL=1

Figure 37-29 shows an example of a 32-byte read to a 32-bit burst enabled port with burst length of 2 words, requiring four 8-byte external transactions. For this case, the address for the 2nd-4th 2-word burst accesses is calculated by adding 0x08 to the lower 5 bits of the 1st address (no carry), and then masking out the lower 3 bits to fix them at zero.

Table 37-7. Examples of 2-word Burst Addresses

1st Address	Lower 5 bits of 1st Address + 0x08 (no carry)	Final 2nd-4th Addresses (After Masking Lower 3 Bits)
0x000	0x08	0x08, 0x10, 0x18
0x008	0x10	0x10, 0x18, 0x00
0x010	0x18	0x18, 0x00, 0x08
0x018	0x00	0x00, 0x8, 0x10
0x020	0x28	0x28, 0x30, 0x38
0x028	0x30	0x30, 0x38, 0x20
0x030	0x38	0x38, 0x20, 0x28
0x038	0x20	0x20, 0x28, 0x30



* For space reasons, the rest of the 4th external access is not shown, but the behavior is identical to the previous 3 burst accesses. Strobe negation timing at end of 4th access is the same as seen in other burst diagrams.

** Timing shown applies when EBI_BR[GCSN]=0. When EBI_BR[GCSN]=1, the subsequent $\overline{TS}$ assertions after the 1st are delayed by 1 cycle each and $\overline{CS}$ negates between each transfer.

*** There is no external $\overline{TA}$ signal. The $\overline{TA}$ signal is generated internally.

Figure 37-31. 32-Byte Read with 4 B-T-B 2-Word Bursts to 32-bit Port, Zero Wait States (SBL=1)

37.4.2.7 Size, Alignment and Packaging on Transfers

Table 37-8 shows the allowed sizes that an internal master can request from the EBI. The behavior of the EBI for request sizes not shown below is undefined. No error signal is asserted for these erroneous cases.

Table 37-8. Transaction Sizes Supported by EBI

# Bytes (internal master)	# Bytes (external master)
1	1
2	2
4	4
3[1]	
8	
16[2]	
32[3]	

1. Some misaligned access cases may result in 3-byte writes. These cases are treated as power-of-2 sized requests by the EBI, using $\overline{\text{WE\_BE}}$ [0:3] to make sure only the appropriate 3 bytes get written.
2. Only supported for case of 32-bit internal AMBA data bus.
3. Only supported for case of 64-bit internal AMBA data bus.

Even though misaligned non-burst transfers from internal masters are supported, the EBI naturally aligns the accesses when it sends them out to the external bus, splitting them into multiple aligned accesses if necessary. See Misaligned Access Support for these cases.

Natural alignment for the EBI means:

- Byte access can have any address
- 16-bit access, address bit 31 must be 0
- 32-bit access, address bits 30-31 must be 0
- For burst accesses of any size, address bits 29-31 must be 0

The EBI never generates a misaligned external access. In the erroneous case that an externally-initiated misaligned access does occur, the EBI errors the access (by asserting $\overline{\text{TEA}}$ externally) and does not initiate the access on the internal bus.

The EBI requires that the portion of the data bus used for a transfer to/from a particular port size be fixed. A 32-bit port must reside on data bus bits 0-31,and a 16-bit port must reside on bits 0-15.

In the following figures and tables the following convention is adopted:

- The most significant byte of a 32-bit operand is OP0, and OP3 is the least significant byte.

- The two bytes of a 16-bit operand are OP0 (most significant) and OP1, or OP2 (most significant) and OP3, depending on the address of the access.
- The single byte of a byte-length operand is OP0, OP1, OP2, or OP3, depending on the address of the access.

This can be seen in Figure 37-32.

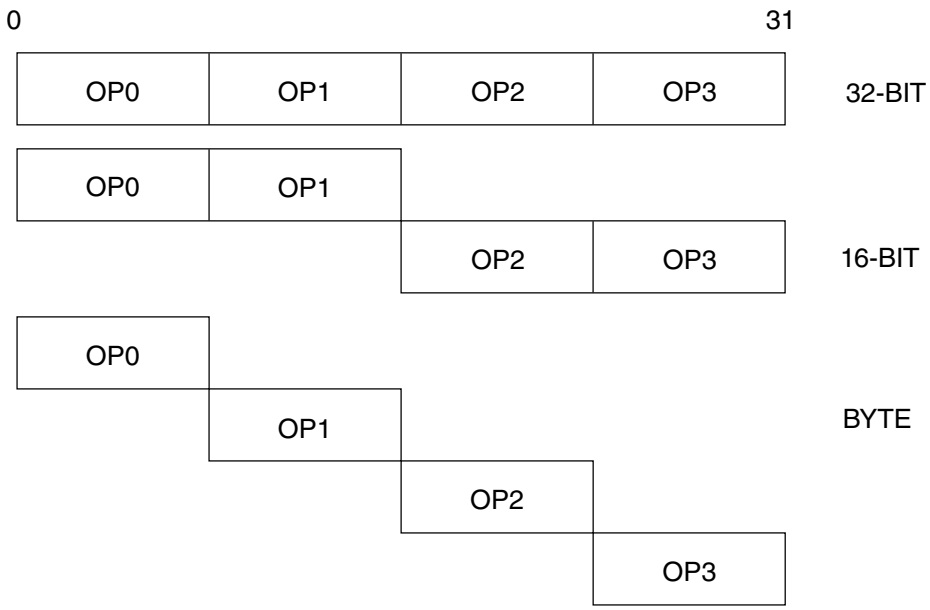


Figure 37-32. Internal Operand Representation

Figure 37-33 shows the device connections on the DATA[0:31] bus.

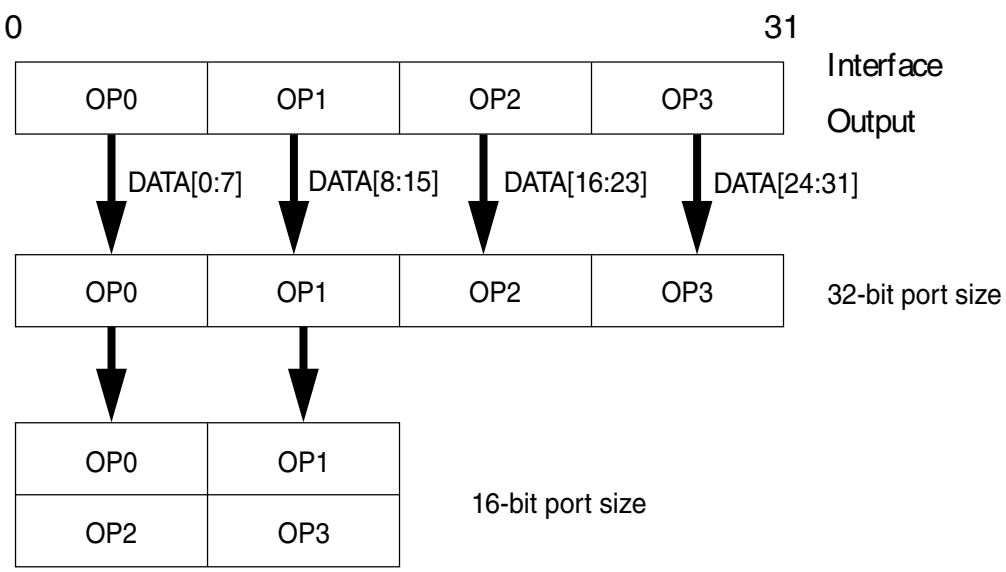


Figure 37-33. Interface to Different Port Size Devices

Table 37-9 lists the bytes required on the data bus for read cycles. The bytes indicated as '-' are not required during that read cycle.

Table 37-10 lists the patterns of the data transfer for write cycles when accesses are initiated by the MCU. The bytes indicated as '-' are not driven during that write cycle.

Table 37-9. Data Bus Requirements for Read Cycles

Transfer Size	EBI_MCR [SIZE]	Address		32-Bit Port Size				16-Bit Port Size[1]	
		A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7[2]	D8:D15[3]
Byte	01	0	0	OP0	-	-	-	OP0	-
	01	0	1	-	OP1	-	-	-	OP1
	01	1	0	-	-	OP2	-	OP2	-
	01	1	1	-	-	-	OP3	-	OP3
16-bit	10	0	0	OP0	OP1	-	-	OP0	OP1
	10	1	0	-	-	OP2	OP3	OP2	OP3
32-bit	00	0	0	OP0	OP1	OP2	OP3	OP0/OP2[4]	OP1/OP3

1. Also applies when DBM=1 for 16-bit data bus mode.
2. For address/data muxed transfers, DATA[16:23] are used externally, not DATA[0:7].
3. For address/data muxed transfers, DATA[24:31] are used externally, not DATA[8:15].
4. This case consists of two 16-bit external transactions, the first writing OP0 and OP1, the second writing OP2 and OP3.

Table 37-10. Data Bus Contents for Write Cycles

Transfer Size	EBI_MCR [SIZE]	Address		32-Bit Port Size				16-Bit Port Size[1]	
		A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7[2]	D8:D15[3]
Byte	01	0	0	OP0	-	-	-	OP0	-
	01	0	1	-	OP1	-	-	-	OP1
	01	1	0	-	-	OP2	-	OP2	-
	01	1	1	-	-	-	OP3	-	OP3
16-bit	10	0	0	OP0	OP1	-	-	OP0	OP1
	10	1	0	-	-	OP2	OP3	OP2	OP3
32-bit	00	0	0	OP0	OP1	OP2	OP3	OP0/OP2[4]	OP1/OP3

1. Also applies when DBM=1 for 16-bit data bus mode.
2. For address/data muxed transfers, DATA[16:23] are used externally, not DATA[0:7].
3. For address/data muxed transfers, DATA[24:31] are used externally, not DATA[8:15].
4. This case consists of two 16-bit external transactions, the first writing OP0 and OP1, the second writing OP2 and OP3.

37.4.2.8 Termination Signals Protocol

The termination signals protocol was defined in order to avoid electrical contention on lines that can be driven by various sources. In order to do that, a slave must not drive signals associated with the data transfer until the address phase is completed and it

recognizes the address as its own. The slave must disconnect from signals immediately after it acknowledges the cycle and not later than the termination of the next address phase cycle.

37.4.2.9 Misaligned Access Support

This section describes all the misaligned cases supported by the EBI. These cases are a subset of the full set of cases allowed by the AMBA AHB V6 specification. The EBI works under the assumption that all internal masters on the device do not produce any misaligned access cases (to the EBI) other than the ones below.

37.4.2.9.1 Misaligned Access Support (64 bit AMBA)

Table 37-11 shows all the misaligned access cases supported by the EBI (using a 64-bit AMBA implementation), as seen on the internal master AMBA bus. All other misaligned cases are not supported. If an unsupported misaligned access to the EBI is attempted (such as non-chip-select or burst misaligned access), the EBI errors the access on the internal bus and does not start the access externally.

Table 37-11. Misalignment Cases Supported by a 64 bit AMBA EBI (internal bus)

No. [1]	Program Size and byte offset	Address [29:31] [2]	Data Bus Byte Strobes [3]	HSIZE [4]	HUNALIGN [5]
1	Half @0x1	001	0110_0000	10	1
2	Half @0x3	011	0001_1000	11 [6]	1
3	Half @0x3	011	0001_0000	01 [7]	1
-	(2 AHB transfers)	100	0000_1000	00	0
4	Half @0x5	101	0000_0110	10	1
5	Half @0x7	111	0000_0001	01 [7]	1
-	(2 AHB transfers)	000	1000_0000	00	0
6	Word @0x1	001	0111_1000	11	1
7	Word @0x1	001	0111_0000	10	1
-	(2 AHB transfers)	100	0000_1000	00	0
8	Word @0x2	010	0011_1100	11 [6]	1
9	Word @0x2	010	0011_0000	10 [8] 01	1
-	(2 AHB transfers)	100	0000_1100		0
10	Word @0x3	011	0001_1110	11	1
11	Word @0x3	011	0001_0000	10 [7] 10	1
12	(2 AHB transfers)	100	0000_1110		1
13	Word @0x5	101	0000_0111	10	1
-	(2 AHB transfers)	000	1000_0000	00	0

Table continues on the next page...

Table 37-11. Misalignment Cases Supported by a 64 bit AMBA EBI (internal bus) (continued)

No. [1]	Program Size and byte offset	Address [29:31] [2]	Data Bus Byte Strobes [3]	HSIZE [4]	HUNALIGN [5]
14	Word @0x6	110	0000_0011	10 [8]	1
-	(2 AHB transfers)	000	1100_0000	01	0
15	Word @0x7	111	0000_0001	10 [7]	1
16	(2 AHB transfers)	000	1110_0000	10	1
17	Doubleword @0x1	001	0111_1111	11	1
-	(2 AHB transfers)	000	1000_0000	00	0
18	Doubleword @0x2	010	0011_1111	11	1
-	(2 AHB transfers)	000	1100_0000	01	0
19	Doubleword @0x3	011	0001_1111	11	1
20	(2 AHB transfers)	000	1110_0000	10	1
21	Doubleword @0x4	100	0000_1111	11 [9]	1
-	(2 AHB transfers)	000	1111_0000	10	0
22	Doubleword @0x5	101	0000_0111	11 [9]	1
23	(2 AHB transfers)	000	1111_1000	11	1
24	Doubleword @0x6	110	0000_00111	11 [8]	1
25	(2 AHB transfers)	000	111_1100	11	1
26	Doubleword @ 0x7	111	0000_0001	11 [7]	1
27	(2 AHB transfers)	000	1111_1110	11	1

1. Misaligned case number. Only transfers where HUNALIGN=1 are numbered as misaligned cases.
2. Address on internal master AHB bus, not necessarily address on external ADDR pins.
3. Internal byte strobe signals on AHB bus. Shown with Big-Endian byte ordering in this table, even though internal master AHB bus uses Little-Endian byte-ordering (EBI flips order internally).
4. Internal signal on AHB bus; 00=8-bits, 01=16 bits, 10=32 bits,11=64-bits. HSIZE is driven according to the smallest aligned container that contains all the requested bytes. This results in extra EBI external transfers in some cases.
5. Internal signal on AHB bus that indicates that this transfer is misaligned (when 1).
6. For this case, the EBI internally treats HSIZE as 10 (4-byte access), if PS=1 (16-bit port).
7. For this case, the EBI internally treats HSIZE as 00(1-byte access).
8. For this case, the EBI internally treats HSIZE as 01 (2-byte access).
9. For this case, the EBI internally treats HSIZE as 10 (4-byte access).

Table 37-12 shows which external transfers are generated by the EBI for the misaligned access cases in Table 37-11, for each port size.

The number of external transfers for each internal AHB master request is determined by the HSIZE value for that request relative to the port size. For example, a half-word write to @011 (misaligned case #2) with 16-bit port size results in 4 external 16-bit transfers because the HSIZE is 64-bits. For cases where two or more external transfers are required for one internal transfer request, these external accesses are considered part of a "small access" set, as described in Small Accesses (Small Port Size and Short Burst Length).

Since all transfers are aligned on the external bus, normal timing diagrams and protocol apply.

Table 37-12. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus)

No.[1]	PS[3]	Program Size and byte offset	ADDR[29:31][4]	$\overline{\text{WE\_BE}}$[0:3][-1]
1	0	Half @0x1	000	1001
	1		000	1011
			010	0111
2	0	Half @0x3	000	1110
			100	0111
	1		010	1011
			100	0111
3	0	Half @0x3	011	1110
-		(2 AHB transfers)	100	0111
3	1		011	1011
-			100	0111
4	0	Half @0x5	100	1001
	1		100	1011
			110	0111
5	0	Half @0x7	111	1110
-		(2 AHB transfers)	000	0111
5	1		111	1011
-			000	0111
6	0	Word @0x1	000	1000
			100	0111
	1		000	1011
			010	0011
			100	0111
7	0	Word @0x1	000	1000
-		(2 AHB transfers)	100	0111
7	1		000	1011
			010	0011
-			100	0111
8	0	Word @0x2	000	1100
			100	0011
	1		010	0011
			100	0011
9	0	Word @0x2	000	1100
-		(2 AHB transfers)	100	0011
9	1		010	0011

Table continues on the next page...

Table 37-12. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus) (continued)

No.[-1]	PS[3]	Program Size and byte offset	ADDR[29:31][4]	$\overline{\text{WE\_BE}}$[0:3][-1]
-			100	0011
10	0	Word @0x3	000	1110
			100	0001
	1		010	1011
			100	0011
			110	0111
11	0	Word @0x3	000	1110
12		(2 AHB transfers)	100	0001
11	1		010	1011
12			100	0011
			110	0111
13	0	Word @0x5	100	1000
-		(2 AHB transfers)	000	0111
13	1		100	1011
			110	0011
-			000	0111
14	0	Word @0x6	110	1100
-		(2 AHB transfers)	000	0011
14	1		110	0011
-			000	0011
15	0	Word @0x7	111	1110
16		(2 AHB transfers)	000	0001
15	1		111	1011
16			000	0011
			010	0111
17	0	Doubleword @0x1	000	1000
		(2 AHB transfers)	100	0000
-			000	0111
17	1		000	1011
			010	0011
			100	0011
			110	0011
-			000	0111
18	0	Doubleword @0x2	000	1100
		(2 AHB transfers)	100	0000
-			000	0011
18	1		010	0011

Table continues on the next page...

Table 37-12. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus) (continued)

No.[-1]	PS[3]	Program Size and byte offset	ADDR[29:31][4]	$\overline{\text{WE\_BE}}$[0:3][-1]
			100	0011
			110	0011
-			000	0011
19	0	Doubleword @0x3	000	1110
		(2 AHB transfers)	100	0000
20			000	0001
19	1		010	1011
			100	0011
			110	0011
20			000	0011
			010	0111
21	0	Doubleword @0x4	100	0000
-		(2 AHB transfers)	000	0000
21	1		100	0011
			110	0011
-			000	0011
			010	0011
22	0	Doubleword @0x5	100	1000
23		(2 AHB transfers)	000	0000
			100	0111
22	1		100	1011
			110	0011
23			000	0011
			010	0011
			100	0111
24	0	Doubleword @0x6	110	1100
25		(2 AHB transfers)	000	0000
			100	0011
24	1		110	0011
25			000	0011
			010	0011
			100	0011
26	0	Doubleword @0x7	111	1110
27		(2 AHB transfers)	000	0000
			100	0001
26	1		111	1011
27			000	0011

Table 37-12. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus)

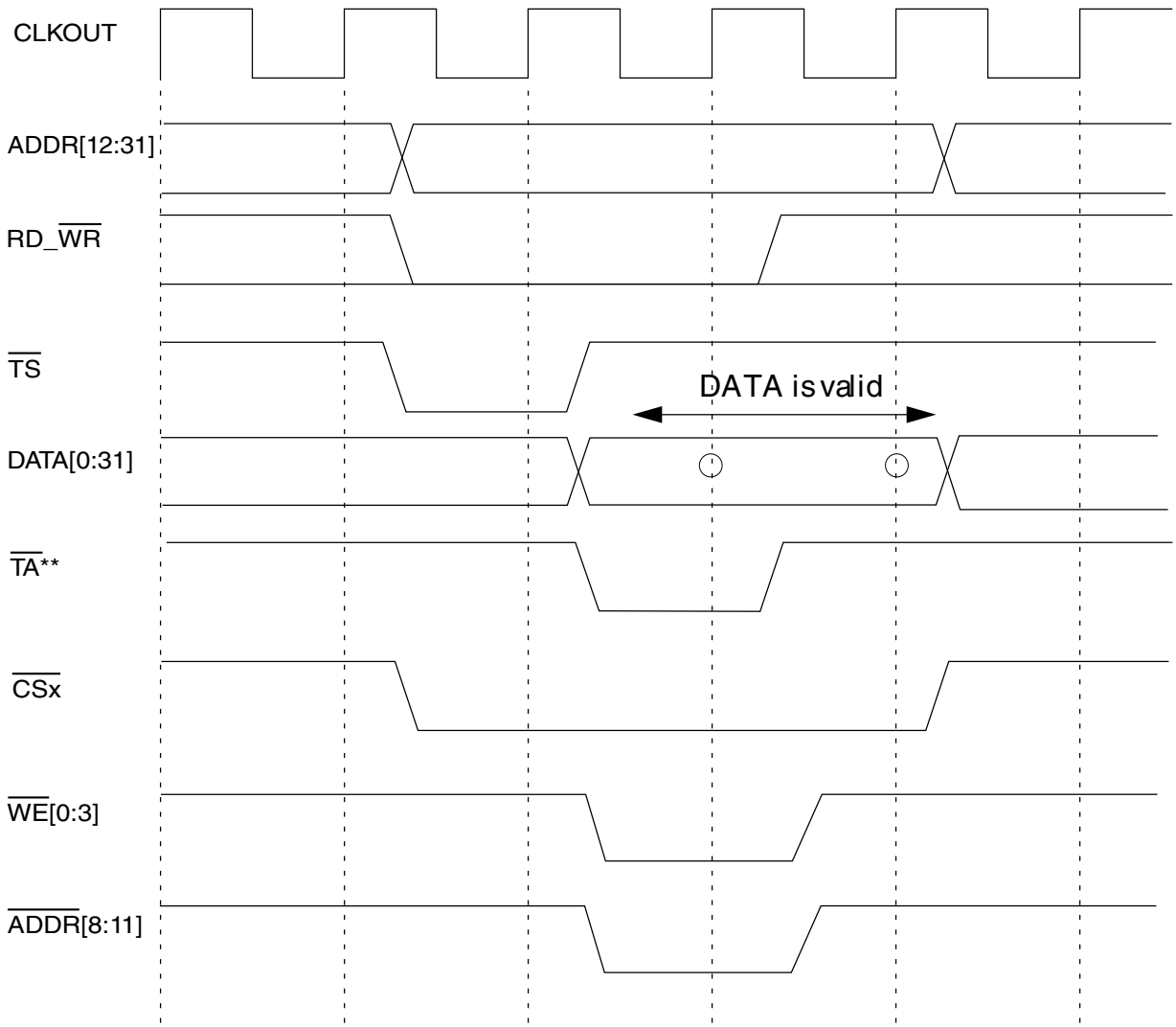
No.[1]	PS[3]	Program Size and byte offset	ADDR[29:31][4]	$\overline{\text{WE\_BE}}$[0:3][5]
			010	0011
			100	0011
			110	0111

1. Misaligned case number, from Table 1.
2. Misaligned case number, from Table 1.
3. Port size; 0=32 bits, 1=16 bits.
4. External ADDR pins, not necessarily the address on internal master AHB bus.
5. External $\overline{\text{WE\_BE}}$ pins. Note that these pins have negative polarity, opposite of the internal byte strobes in Table 1.

37.4.2.10 Address / Write Enable Multiplexing

Address/Write Enable multiplexing enables the design of a system with reduced pin count (saves 4 pins). The typical usage is the case where the SoC is connected to an external chip-select device that requires write-enable functionality, but does not require the full address bus width. In such a system, the user can set EBI_OR[AWE]=1 to cause the EBI to drive its internal WE[0:3] signals onto the upper 4 ADDR pins, instead of relying on separate WE[0:3] pins (which may not be present on some SoCs).

In general, timing diagrams using ADDR/WE multiplexing (EBI_OR[AWE]=1) are very similar to other diagrams in this document, except for the behavior of the upper 4 ADDR signals, which can be seen in Figure 37-34.



* The upper 4 address bits just mirror WE[0:3].
** There is no external TA signal. The TA signal is generated internally

Figure 37-34. Single Beat 32-bit Write Cycle, CS Access, Zero Wait States, AWE=1

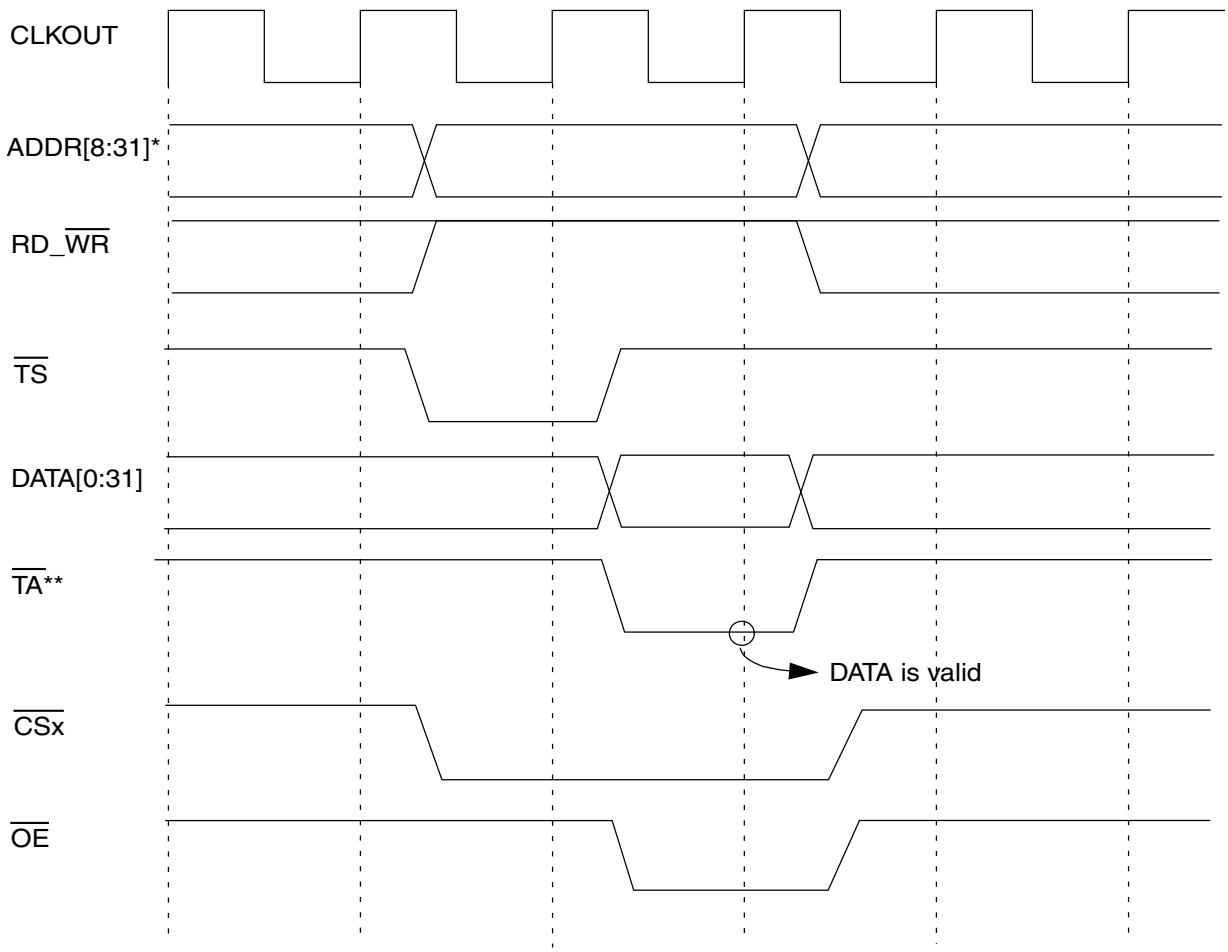
37.4.2.11 Address by Port Size

Address by Port Size (EBI_OR[APS]=1) provides a way to increase the memory size supported by the EBI for external chip-select devices with 16 or 32-bit port size.

For example, consider an SoC that has ADDR[8:31] pinned out and connects to an external 32-bit chip-select memory:

- If APS=0, then ADDR[31] is not connected to the memory and thus only 22 address lines (ADDR[8:29]) are left to connect to the memory.
- If APS=1, then the EBI drives internal address 6:29 to external ADDR[8:31], thus giving the user a 24-bit address range to access the 16-bit memory.

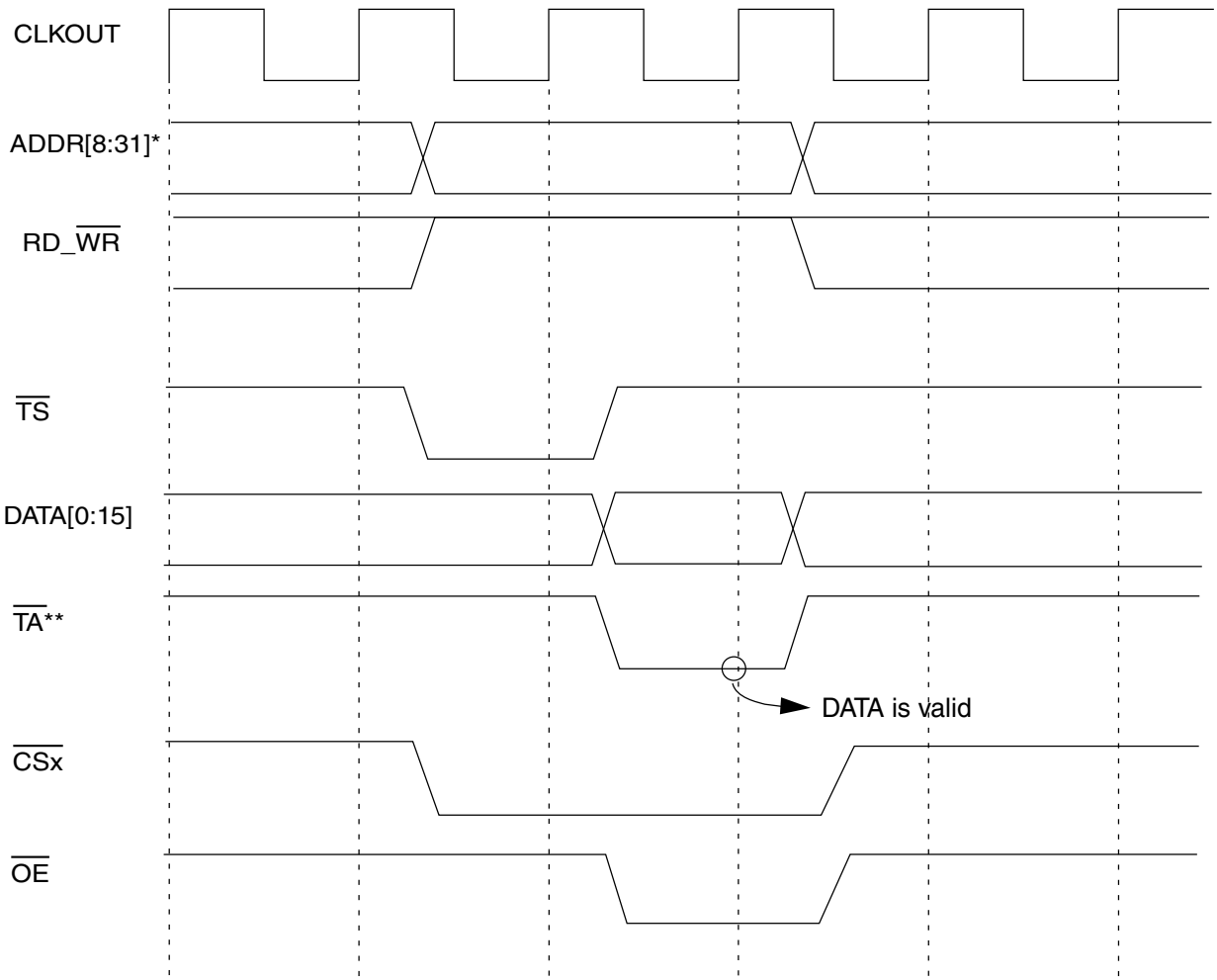
In general, timing diagrams using APS=1 are very similar to other diagrams in this document, except for the shifting of the ADDR signals, which can be seen in Figure 37-35 and Figure 37-36.



* Only difference from APS=0 timing is ADDR[8:31] is driven by internal address 6:29 for this case. If the full ADDR[3:31] bus is being pinned out, then the EBI drives ADDR[3:31] with internal address 1:29

**There is no external TA signal. The TA signal is generated internally.

Figure 37-35. Single Beat 32-bit Read Cycle, CS Access, Zero Wait States, APS=1, PS=0



* Only difference from APS=0 timing is ADDR[8:31] is driven by internal address 7:30 in this case. If the full ADDR[3:31] range is being used, then the EBI drives ADDR[3:31] with internal address 2:30

**There is no external TA signal. The TA signal is generated internally.

Figure 37-36. Single Beat 16-bit Read Cycle, CS Access, Zero Wait States, APS=1, PS=1

37.4.2.12 32-bit AMBA Burst on 64-bit AMBA Case

This section only applies for those SoC's that use the EBI with an AMBA bus width of 64 bits. This section is N/A for those SoC's that use the EBI in a 32-bit AMBA configuration (determined by SoC parameter).

On some SoCs that use this EBI, the internal AMBA master is capable of requesting a WRAP4 32-bit burst even with a 64-bit AMBA bus width. This case is supported by the AMBA standard, but is rarely used on SoCs, since normally the full AMBA width is used

when bursting (for optimum performance). A scenario where this case occurs is when there is a 32-bit AMBA master connecting through a 32->64 bit gasket that drives requests to the EBI's 64-bit AMBA bus.

In the 32-bit burst on 64-bit AMBA case, the EBI returns data on the appropriate half of the 64-bit AMBA bus, based on the address of the transfer (address bit 2 using N:0 numbering), and then alternates halves on each AMBA burst beat. Examples below are based on AMBA data bus numbering 63:0.

- For a burst starting at address 0x0 (or 0x8, 0x10, etc.), the 4 beats of data are returned on hrdata [63:32], [31:0], [63:32], [31:0].
- For a burst starting at address 0x4 (or 0xC, 0x14, etc.), the 4 beats of data are returned on hrdata [31:0], [63:32], [31:0], [63:32]

Note

For the case of a 32-bit burst on 64-bit AMBA where EBI_BR[BL]=0 and EBI_BR[SBL]=0, the external burst size (8 words) is mismatched to the internal burst size (4 words). In this case, to avoid erroneous data due to different wrapping boundaries, the EBI disables external burst for that access and splits the internal burst request to multiple single external accesses.

37.4.2.13 Internal AMBA Master Burst Read Abort Case

On some SoCs that use this EBI, the internal AMBA master has the capability to abort the AMBA burst read early, without performing the 4 requests that are normally part of an AMBA WRAP4 burst sequence. For example, instead of the usual htrans=NSEQ, SEQ, SEQ, SEQ, the master might do 1 of these cases:

- htrans=NSEQ (with hburst=WRAP4), SEQ (0-2 times), NSEQ (2nd NSEQ aborts prior burst)
- htrans=NSEQ (with hburst=WRAP4), SEQ (0-2 times), IDLE (IDLE aborts prior burst)

In these cases, the EBI has already started the burst externally, but will abort it early (without asserting internal hresp), and only returns the beats of data on the internal AMBA bus corresponding to the AMBA NSEQ/SEQ requests received. In a subset of

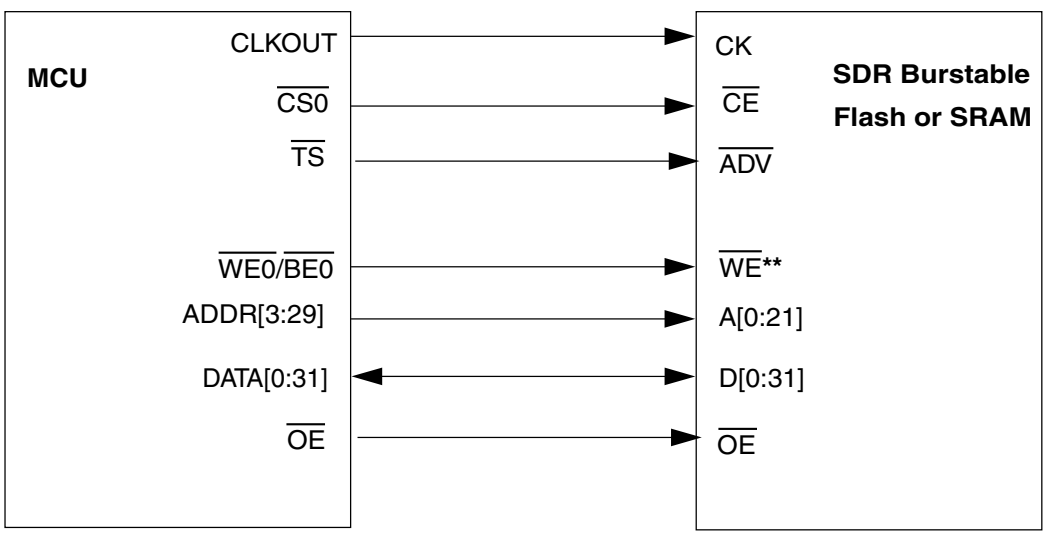
these master burst abort cases, there may be extra $\overline{TS}$ /$\overline{TA}$ transfers on the external bus after the internal AMBA master has aborted (e.g. finishing a "small access" set of transfers). These extra read accesses are extraneous and can just be ignored.

This feature is not part of the AMBA standard, but is implemented in some AMBA masters for performance reasons. The master burst abort feature applies to reads only. The behavior of the EBI for an aborted AMBA burst write is undefined.

37.5 Initialization/Application Information

37.5.1 Running with SDR (Single Data Rate) Burst Memories

This includes FLASH and SRAM memories with a compatible burst interface. Figure 37-37 shows a block diagram of an MCU connected to a 32-bit SDR burst memory.



* May or may not be connected, depending on the memory used.

** Flash memories typically use one $\overline{WE}$ signal as shown, RAMs use 2 or 4 (16-bit or 32-bit)

Figure 37-37. MCU Connected to SDR Burst Memory

Refer to Figure 37-24 for an example of the timing of a typical Burst Read operation to an SDR burst memory. Refer to Figure 37-9 for an example of the timing of a typical Single Write operation to SDR memory.

37.5.2 Running with Asynchronous Memories

The EBI also supports asynchronous memories. In this case, the CLKOUT and $\overline{TS}$ pins are not used by the memory and bursting is not supported. However, the EBI still drives these outputs, and always drives and latches all signals at posedge CLKOUT (i.e., there is no "asynchronous mode" for the EBI). The data timing is controlled by setting the SCY bits in the appropriate Option Register to the proper number of wait states to work with the access time of the asynchronous memory, just as done for a synchronous memory.

37.5.2.1 Example Wait State Calculation

This example applies to any chip-select memory, synchronous or asynchronous.

As an example, say we have a memory with 50ns access time, and we are running the external bus @ 66 MHz (CLKOUT period: 15.2ns). Assume the input data spec for the MCU is 4ns.

number of wait states = (access time) / (CLKOUT period) + (0 or 1) (depending on setup time)

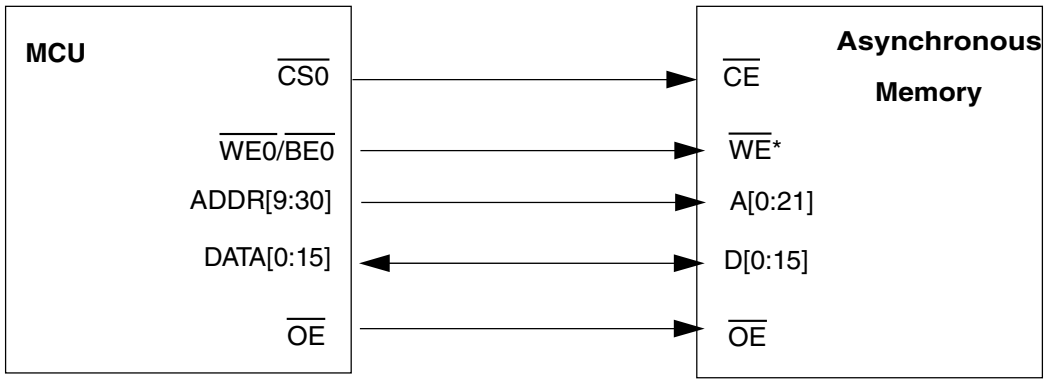
50/15.2 = 3 with 4.4ns remaining (so we need at least 3 wait states, now check setup time)

15.2-4.4=10.8ns (this is the achieved input data setup time)

Since actual input setup (10.8ns) is greater than the input setup spec (4.0ns), 3 wait states is sufficient. If the actual input setup was less than 4.0ns, we would have to use 4 wait states instead.

37.5.2.2 Timing and Connections for Asynchronous Memories

The connections to an asynchronous memory are the same as for a synchronous memory, except that the CLKOUT and $\overline{TS}$ signals are not used. Figure 37-38 shows a block diagram of an MCU connected to an asynchronous memory.

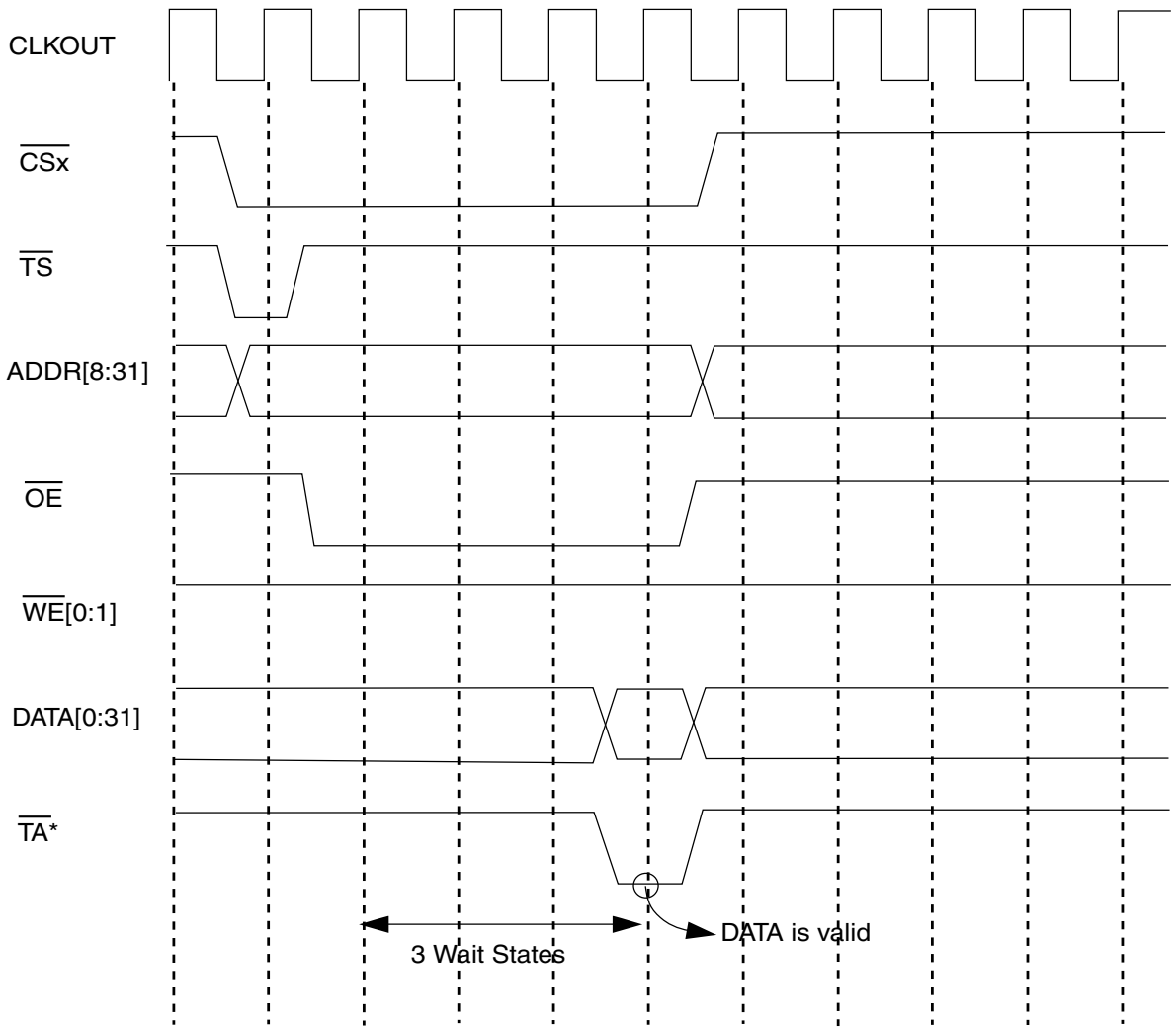


* Flash memories typically use one WE signal as shown, RAMs use 2 or 4 (16-bit or 32-bit)

Figure 37-38. MCU Connected to Asynchronous Memory

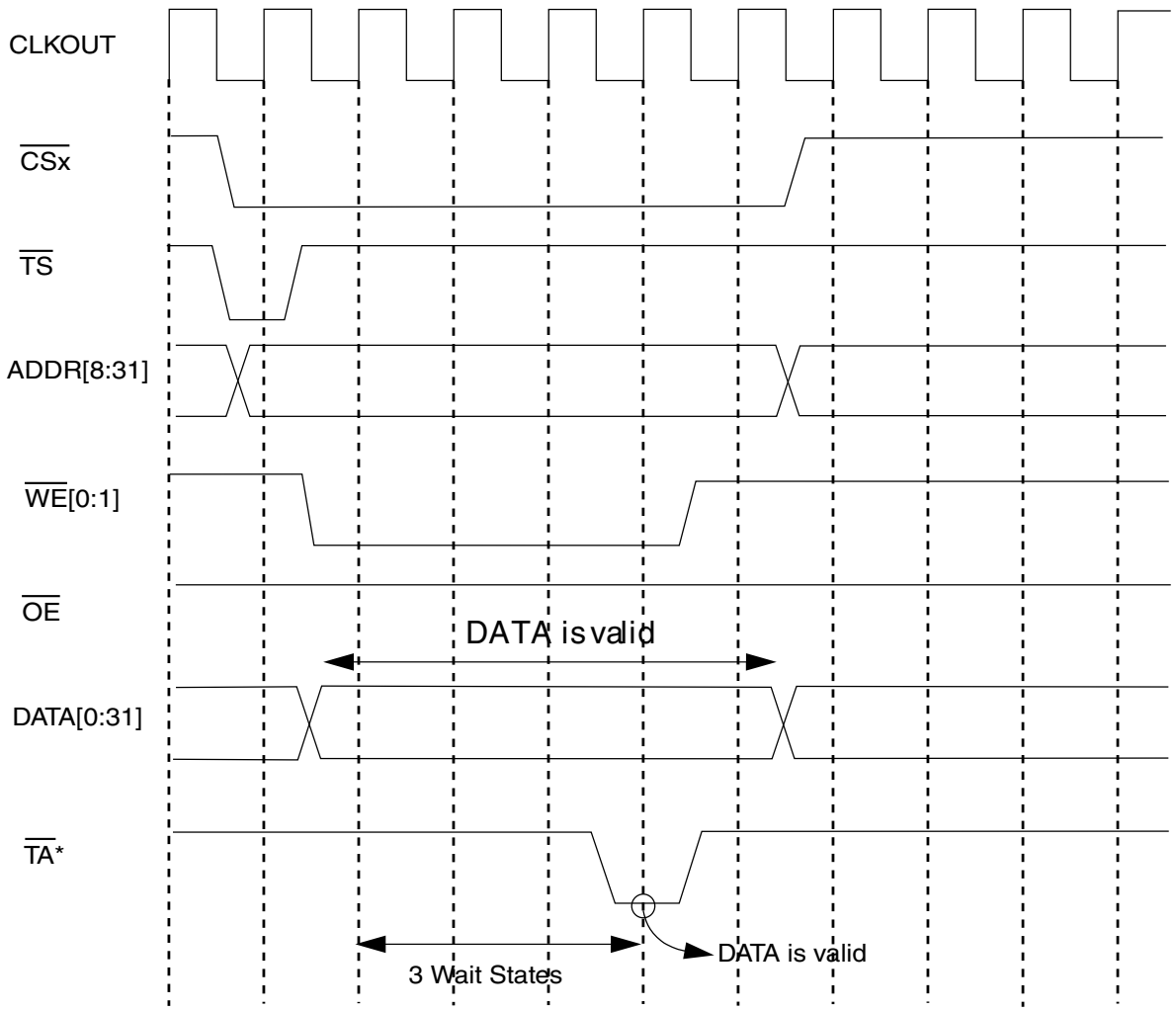
Figure 37-39 shows a timing diagram of a read operation to a 16-bit asynchronous memory using 3 wait states.

Figure 37-40 shows a timing diagram of a write operation to a 16-bit asynchronous memory using 3 wait states.



*There is no external TA signal. The TA signal is generated internally.

Figure 37-39. Read Operation to Asynchronous Memory, Three Initial Wait States



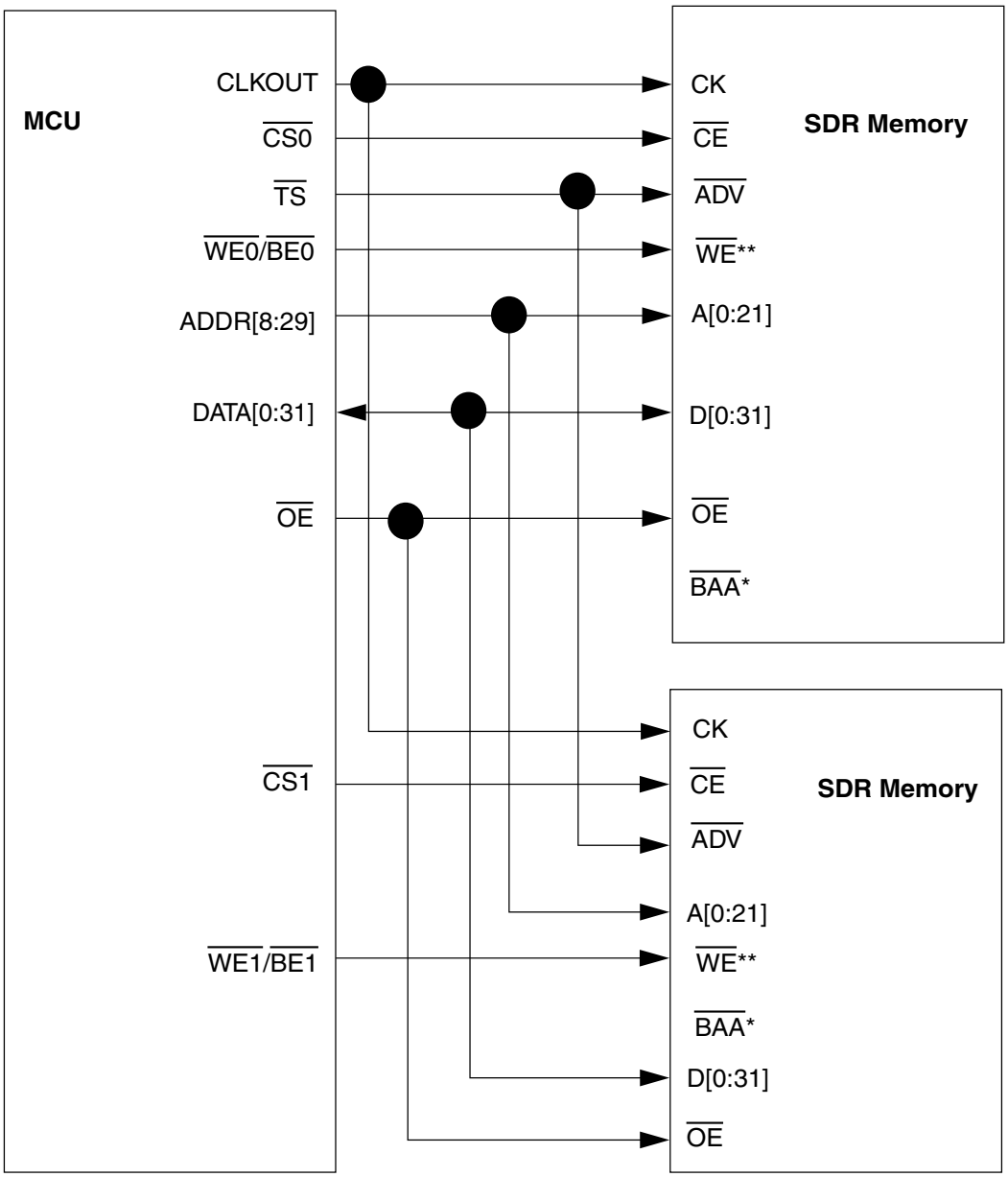
*There is no external TA signal. The TA signal is generated internally.

Figure 37-40. Write Operation to Asynchronous Memory, Three Initial Wait States

37.5.3 Connecting an MCU to Multiple Memories

The MCU can be connected to more than one memory at a time.

Figure 37-41 shows an example of how two memories could be connected to one MCU.



* May or may not be connected, depending on the memory used.

** Flash memories typically use one $\overline{WE}$ signal as shown, RAMs use 2 or 4 (16-bit or 32-bit)

Figure 37-41. MCU Connected to Multiple Memories

Chapter 38
Analog-to-Digital Converters (ADC) Configuration

38.1 ADC overview

This chapter is organized into the following sections:

- ADC overview
  - ADC subsystem block diagram
  - Analog input pin multiplexing
- Configuration of ADC modules
  - Sigma-Delta Analog-to-Digital Converter (SDADC)
  - Successive Approximation Register Analog-to-Digital Converter (SARADC)

MPC5777M contains the following analog-to-digital converters (ADCs):

- 12 independent 12-bit Successive Approximation Register Analog-to-Digital Converters (SARADCs)
- 10 independent 16-bit Sigma-Delta Analog-to-Digital Converters (Sigma-Delta, or SDADCs)

These ADCs interface to external analog input pins:

Number of external analog input pins	Package
60	416 BGA
84	512 BGA

Each ADC instance, whether SAR or Sigma-Delta, has independent analog input pin multiplexing, register interface, trigger and clock inputs, independent interrupt, and DMA capability. Each SAR and Sigma-Delta ADC has a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

The 416/512 BGA packages have separate supply and ground pins for the SAR and the Sigma-Delta ADCs:

- VDD_HV_ADV_S/VSS_HV_ADV_S for SAR
- VDD_HV_ADV_D/VSS_HV_ADV_D for Sigma-Delta

The SARADCs on the device are referred to as SARADC_B (84:1 input mux), SARADC_0–SARADC_10 (maximum 8:1 input mux for each).

38.1.1 ADC subsystem block diagram

The following figure gives the block diagram for the high-level integration of the ADC sub-system for MPC5777M. The primary intent of the diagram is to show that each ADC, whether SAR or Sigma-Delta, have independent register interfaces, interrupt/DMA requests, and trigger source selection.

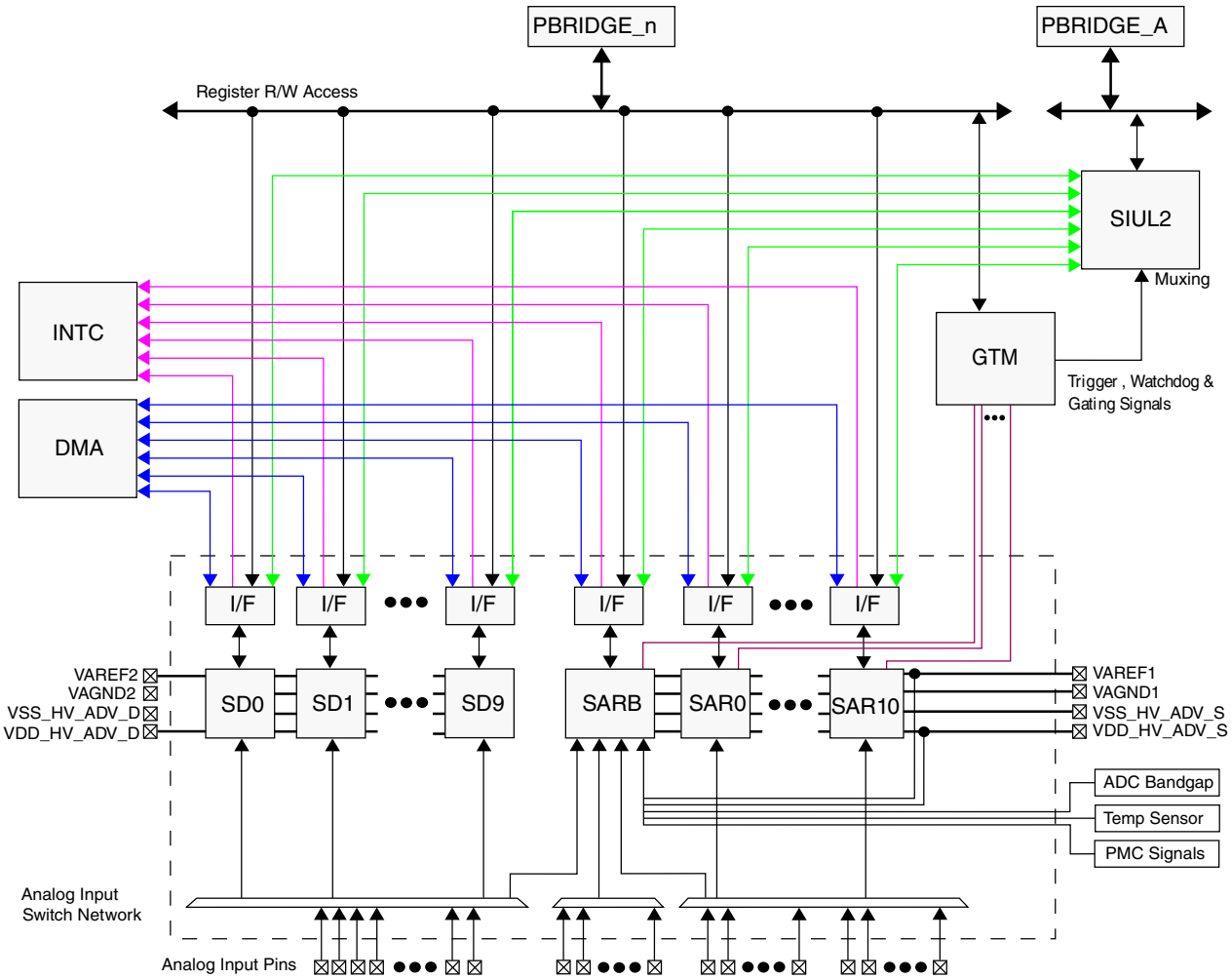


Figure 38-1. MPC5777M ADC subsystem

38.1.2 Analog input pin multiplexing

There are 60 external analog input pins in the 416 BGA package for MPC5777M, and 84 in the 512 BGA package. All analog input pins in either package are readable by SARADC_B. In addition to SARADC_B, all analog input pins are routed to either the Sigma-Delta or SARADCs, but never both. The following table gives the analog input pin assignment for the packages.

Table 38-1. MPC5777M analog input pin multiplexing

ADC instance	416 BGA		512 BGA	
	Analog input pins	Mux input[1]	Analog input pins	Mux input[1]
SARADC_B	AN0:AN11, AN16:AN21, AN24:AN29, AN32:AN33, AN36:AN37, AN40:AN43, AN48:AN53, AN56:AN61, AN64:AN65, AN68:AN73,	0:11, 16:21, 24:29, 32:33, 36:37, 40:43, 48:65, 68:73,	AN0:AN87	87

Table continues on the next page...

Table 38-1. MPC5777M analog input pin multiplexing (continued)

ADC instance	416 BGA		512 BGA	
	Analog input pins	Mux input[1]	Analog input pins	Mux input[1]
	AN76:AN77, AN80:AN81, AN84:AN87	76:77, 80:81, 84:87		
SARADC_0	AN4:AN7	4:7	AN4:AN7	4:7
SARADC_1	AN8:AN11	8:11	AN8:AN15	8:15
SARADC_2	AN16:AN19	16:19	AN16:AN19, AN22:AN23	16:19, 22:23
SARADC_3	AN26:AN29	26:29	AN26:AN31	26:31
SARADC_4	AN32:AN33	32:33	AN32:AN35	32:35
SARADC_5	AN40:AN43	40:43	AN40:AN43	40:43
SARADC_6	AN48:AN51	48:51	AN48:AN51	48:51
SARADC_7	AN56:AN59	56:59	AN56:AN59	56:59
SARADC_8	AN64:AN65	64:65	AN64:AN67	64:67
SARADC_9	AN72:AN73	72:73	AN72:AN75	72:75
SARADC_10	AN80:AN81	80:81	AN80:AN83	80:83
SDADC_0	AN2:AN3	AN[0]:AN[1]	AN2:AN3	AN[0]:AN[1]
SDADC_1	AN0:AN1	AN[0]:AN[1]	AN0:AN1	AN[0]:AN[1]
SDADC_2	AN20:AN21, AN24:AN25	AN[0]:AN[1], AN[2]:AN[3]	AN20:AN21, AN24:AN25	AN[0]:AN[1], AN[2]:AN[3]
SDADC_3	AN36:37	AN[0]:AN[1]	AN36:AN39, AN44:AN47	AN[0]:AN[7]
SDADC_4	AN52:AN53	AN[0]:AN[1]	AN52:AN53	AN[0]:AN[1]
SDADC_5	AN60:AN61	AN[0]:AN[1]	AN60:AN61	AN[0]:AN[1]
SDADC_6	AN68:AN71	AN[0]:AN[3]	AN68:AN71	AN[0]:AN[1], AN[2]:AN[3]
SDADC_7	AN76:AN77	AN[0]:AN[1]	AN76:AN79	AN[0]:AN[1], AN[2]:AN[3]
SDADC_8	AN84:AN85	AN[0]:AN[1]	AN84:AN85	AN[0]:AN[1]
SDADC_9	AN86:AN87	AN[0]:AN[1]	AN86:AN87	AN[0]:AN[1]

1. For the Sigma-Delta mux inputs, even channel numbers (0,2,4,6) correspond to the positive terminal of the differential input, and odd channel numbers (1,3,5,7) correspond to the negative terminal.

38.2 Configuration of ADC modules

The following sections describe device-specific details for the following modules:

- Sigma-Delta Analog-to-Digital Converter (SDADC)
- Successive Approximation Register Analog-to-Digital Converter (SARADC)

38.2.1 Sigma-Delta Analog-to-Digital Converter (SDADC)

There are ten independent Sigma-Delta ADCs on MPC5777M. Refer to Figure 38-1 for the ADC subsystem block diagram.

The following sections describe configuration details:

- Sigma-Delta ADC integration diagram
- Sigma-Delta ADC analog input multiplexing
- Sigma-Delta ADC clock sources
- Sigma-Delta ADC starting/triggering
- Sigma-Delta ADC external modulator
- Conversion result interrupt/DMA
- Sigma-Delta ADC analog input channel selection

For detailed module information, refer to the Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface chapter.

38.2.1.1 Sigma-Delta ADC integration diagram

The following figure shows the block diagram of the Sigma-Delta ADC integration on MPC5777M. Note in the diagram that any unused Sigma-Delta ADC input can be read by SARADC_B. This is true for the unused input in single-ended operation.

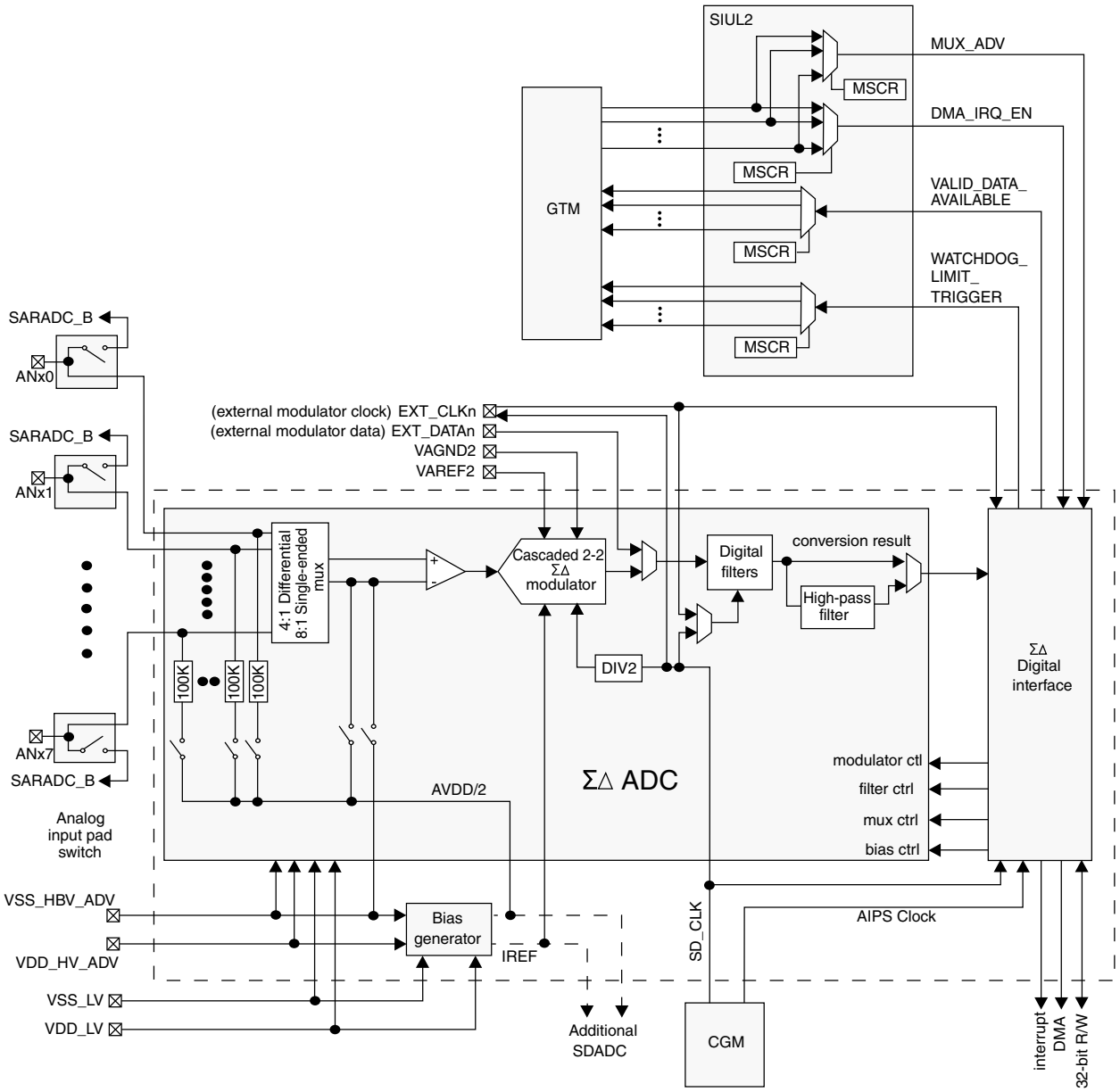


Figure 38-2. Sigma-Delta ADC integration diagram

38.2.1.2 Sigma-Delta ADC analog input multiplexing

The analog input multiplexing switches are contained within the Sigma-Delta ADCs. A single analog switch pad cell is used for the Sigma-Delta analog input pins in order to share the pin with SARADC_B. The connection to the Sigma-Delta ADC inputs is made to the pad pin through the analog switch pad cell, shown in Figure 38-2, and is controlled as follows:

- SARADC_B analog inputs are switchable by the SIUL2_MSCRn[APC] bit.
- SDADC analog inputs have no SIUL2_MSCRn[APC] bit control, but are directly connected to the pad.

The mapping of analog input pins to the Sigma-Delta converters is given in Table 38-1.

38.2.1.3 Sigma-Delta ADC clock sources

Each Sigma-Delta ADC digital interface on the MPC5777M has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the modulator clock (SD_CLK). The input clock for the Sigma-Delta modulator is the SD_CLK. The following figure gives a block diagram of the required Sigma-Delta ADC clock architecture. See the clocking chapter, Sigma-Delta ADC clocking, for more detail on the clock sources.

The frequency range for the SD_CLK is 4–16 MHz. Both the analog and digital blocks of the ADC contain an internal clock mux for external modulator mode. The external modulator clock (EXT_CLK) can be input to the device, or generated from SD_CLK from the CGM. The pad input/output settings in the corresponding SIUL2 MSCR pad control registers determine the external modulator clock source to the ADC blocks. The EXT_CLK pins are multiplexed on multiple external pins on the device. Selection of the EXT_CLK pin to use for the external modulator clock for each ADC is done in the SIUL2 MSCR input multiplexing registers. Each Sigma-Delta ADC has three external pins to select from for an external modulator clock.

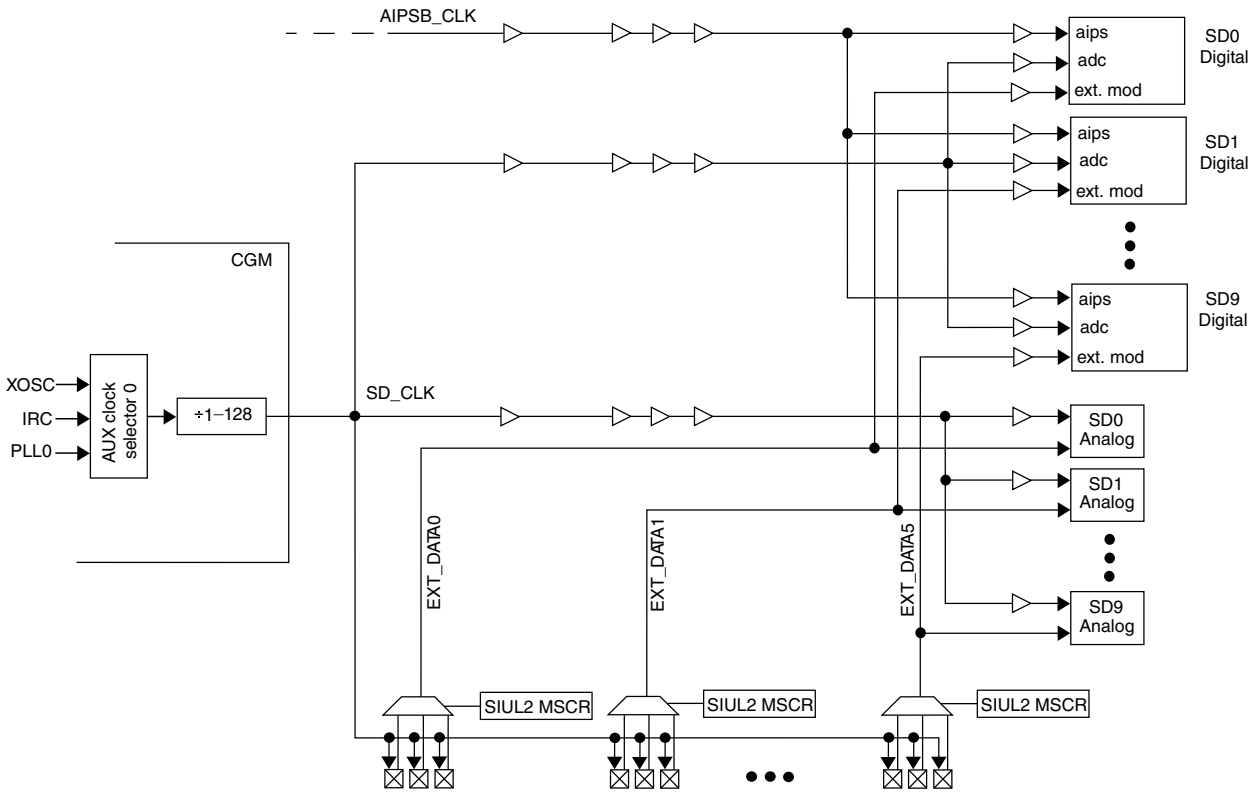


Figure 38-3. Sigma-Delta ADC clock diagram

38.2.1.3.1 Clock Skew

To avoid crosstalk issues, all Sigma-Delta ADCs run at the same clock frequency. Independent programmable oversampling rates are provided to support the varying bandwidths of interest on each ADC in the application. Therefore, all Sigma-Delta analog blocks are clocked from the output of a single divider of the peripheral clock at the chip level.

38.2.1.4 Sigma-Delta ADC starting/triggering

Each Sigma-Delta ADC on MPC5777M supports starting the converter with a software write to an enable bit in a register in the digital interface.

38.2.1.4.1 Simultaneous start of multiple Sigma-Delta converters

A method to start multiple Sigma-Delta converters simultaneously by a single software write to a control register is available on the device. Simultaneously in this case means that all selected converters start sampling their inputs on the same SD_CLK clock edge.

The digital interface of the Sigma-Delta ADC contains the control register for simultaneous start of the ADCs. When the register is written to start multiple converters, an output from the digital interface is asserted. The hardware trigger input of each Sigma-Delta ADC is connected to this simultaneous start output. This way the user controls which ADCs are selected by enabling the hardware trigger of desired ADCs before writing to the starting simultaneous converters.

The connections for the simultaneous start of the converters is given in the following figure.

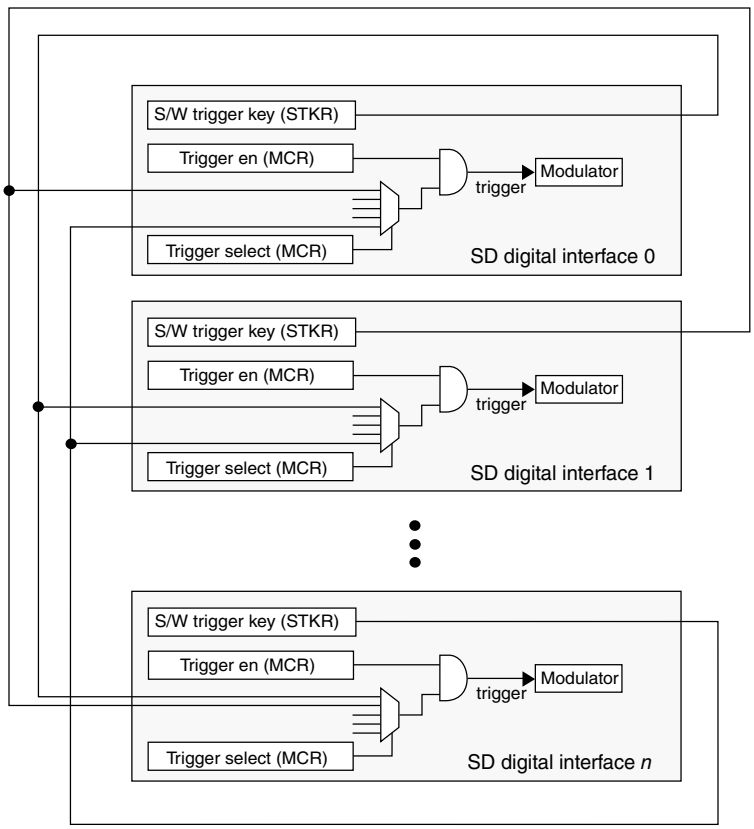


Figure 38-4. Sigma-Delta ADC simultaneous start connectivity

38.2.1.4.2 Sigma-Delta mux input advance and wraparound

The mux wraparound feature is enabled and controlled in the digital interface of the Sigma-Delta ADC. The advancing of the mux can be controlled in the interface by software or by external GTM channels. The mapping of GTM output channels to the mux advance input for each Sigma-Delta ADC for each device is given in the SIUL2 MSCR registers.

38.2.1.5 Sigma-Delta ADC external modulator

38.2.1.5.1 External modulators

The Sigma-Delta ADC supports an interface to external Sigma-Delta modulators. In this mode, the Sigma-Delta ADC internal modulator is disabled, and a serial bit stream from an external modulator is clocked into the back-end digital filters of the on-chip Sigma-Delta ADC. Currently, the AMC1203 external modulator is supported by the Sigma-Delta ADC. It is possible that other external 2nd order modulators with clock rates less than 10 MHz and a compatible electrical interface (see MPC5777M Data Sheet) could be supported.

The external modulator interface for each Sigma-Delta ADC on MPC5777M is shown in the following figure.

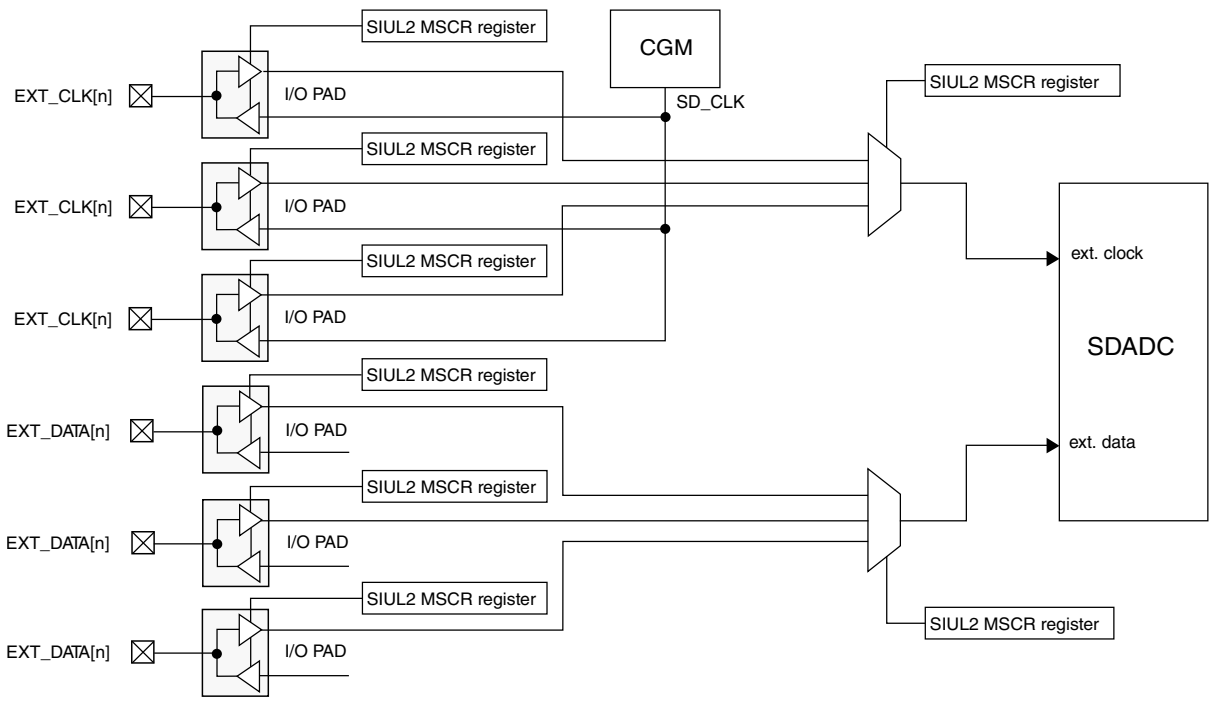


Figure 38-5. Sigma-Delta ADC external modulator interface

38.2.1.5.2 External modulator clock/data inputs

Two external pins are required for each Sigma-Delta ADC to support an external modulator, an input data pin EXT_DATA (bitstream data), and an input/output clock pin EXT_CLK (bitstream clock). When the external modulator is used, the clock and data inputs to the Sigma-Delta ADC digital filters come from the EXT_CLK and EXT_DATA pins, instead of the internal modulator as shown in Figure 38-12. External modulator operation is enabled in the ADC digital interface registers. The internal modulator is

disabled. The EXT_CLK and EXT_DATA inputs are multiplexed on port input pins. For each Sigma-Delta ADC, there are three sets of EXT_CLK/EXT_DATA pins, and selection of those pins for each ADC is done in the SIUL2 MSCR registers.

38.2.1.5.3 External modulator clock output

MPC5777M provides an option for an output clock from the device to the external modulator. When providing an output clock, the SD_CLK from the CGM is connected to the EXT_CLK output pin. Both the input and output buffer of the EXT_CLK pin must be enabled in the corresponding SIUL2 MSCR register, in order for both the external modulator and internal filters to be clocked from the SD_CLK. This is shown in Figure 38-5.

38.2.1.6 Conversion result interrupt/DMA

A DMA or interrupt can be generated for each conversion for each ADC. The Sigma-Delta ADC has an 8-deep result FIFO.

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input signal is connected to outputs from the GTM timer block. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the signal from the GTM is asserted, no interrupt or DMA request is generated for any conversion results. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

The gating signal for each ADC is driven by several GTM channels. See the System Integration Unit Lite2 (SIUL2) chapter for the specific mapping of GTM channels to Sigma-Delta interrupt/DMA gating signals.

38.2.1.7 Conversion Limit Watchdog Output to GTM

Each Sigma-Delta ADC provides an optional conversion limit check for upper and lower bounds. The enable bit and the two 16-bit limit registers for the feature are included in the SDADC wrapper. If a conversion result exceeds one of the limits, a flag bit is asserted in a status register in the SDADC wrapper. The flag bit is output to the GTM for each SDADC. The mapping of conversion limit watchdog outputs to GTM channels is defined in the SIUL2 MSCR registers.

38.2.1.8 SDADC_MCR[TRIGSEL] definitions

For this chip, the following input triggers apply:

Table 38-2. Input triggers for this chip

TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware-triggered conversions. 0000 SDADC_0 0001 SDADC_1 0010 SDADC_2 0011 SDADC_3 0100 SDADC_4 0101 SDADC_5 0110 SDADC_6 0111 SDADC_7 1000 SDADC_8 1001 SDADC_9 1010 – 1110 Reserved 1111 MUX_ADV **Note:** Selecting the input trigger source to be from the output trigger of the same SDADC is not prohibited by hardware, but is not supported, and will result in the SDADC never responding to the input trigger.

38.2.1.9 Sigma-Delta ADC analog input channel selection

For the Sigma-Delta mux inputs AN[x], the positive and negative input terminals are selected in the Sigma-Delta ADC digital interface Channel Selection Register (CSR).

This section shows analog channel selections for each combination of Module Configuration Register (MCR) field values MCR_MODE and MCR_VCOMSEL. The following table provides a quick reference key to the MCR fields used in the SD ADC analog input selection tables that follow it.

Table 38-3. Register fields for SD ADC analog input channel selection

Field	Description
MCR_VCOMSEL	Common Voltage Bias Selection

Table continues on the next page...

Table 38-3. Register fields for SD ADC analog input channel selection (continued)

Field	Description
	This bit selects the common voltage bias for the negative input terminal of SDADC during single-ended mode (MODE=1). 0 Negative input terminal is biased with VREFN. 1 Negative input terminal is biased with VREFP/2 (half scale bias).
MCR_MODE	Mode selection 0 Differential input mode is selected. 1 Single-ended input mode is selected.
CSR_ANCHSEL	Analog Channel Selection Based on single-ended or differential mode of operation (MODE bit of MCR), common mode voltage selection (VCOMSEL bit of MCR), this bit-field defines the connectivity of analog inputs to either positive or negative polarity terminals of Sigma-Delta ADC, as shown in the following tables: • Table 38-4 for SDADC_0 • Table 38-5 for SDADC_1 • Table 38-6 for SDADC_2 • Table 38-7 for SDADC_3 • Table 38-8 for SDADC_4 • Table 38-9 for SDADC_5 • Table 38-10 for SDADC_6 • Table 38-11 for SDADC_7 • Table 38-12 for SDADC_8 • Table 38-13 for SDADC_9

Table 38-4. SDADC_0 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	

Table continues on the next page...

Table 38-4. SDADC_0 analog input AN[x] selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-5. SDADC_1 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-6. SDADC_2 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-7. SDADC_3 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
	1	000	AN[0]	VREFP/2
		001	AN[1]	

Table continues on the next page...

Table 38-7. SDADC_3 analog input AN[x] selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	AN[4]	AN[5]
		011	AN[6]	AN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-8. SDADC_4 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	

Table continues on the next page...

Table 38-8. SDADC_4 analog input AN[x] selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-9. SDADC_5 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-10. SDADC_6 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN

Table continues on the next page...

Table 38-10. SDADC_6 analog input AN[x] selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-11. SDADC_7 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	

Table continues on the next page...

Table 38-11. SDADC_7 analog input AN[x] selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-12. SDADC_8 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN

Table continues on the next page...

Table 38-12. SDADC_8 analog input AN[x] selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 38-13. SDADC_9 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

38.2.2 Successive Approximation Register Analog-to-Digital Converter (SARADC)

There are twelve independent SARADCs on MPC5777M. Refer to Figure 38-1 for the ADC sub-system block diagram.

The following sections describe configuration details:

- SAR ADC integration diagram
- SAR ADC analog input pin multiplexing
- SAR ADC clock sources
- SAR ADC triggering
- SAR ADC alternate reference
- SAR absolute voltage reference
- SAR ADC diagnostic
- SARADC channel assignment
- SAR ADC register definitions

For detailed module information, refer to the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

38.2.2.1 SAR ADC integration diagram

The following figure shows the block diagram of the integration of the twelve ADCs on MPC5777M.

SARADC_B implements a bias generator converting the VDD_HV_ADR_S, 1/3 VDD_HV_ADR_S, 2/3 VDD_HV_ADR_S, and VSS_HV_ADR_S reference points through the internal 20 kΩ source impedance. See the SARADC block diagram in the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

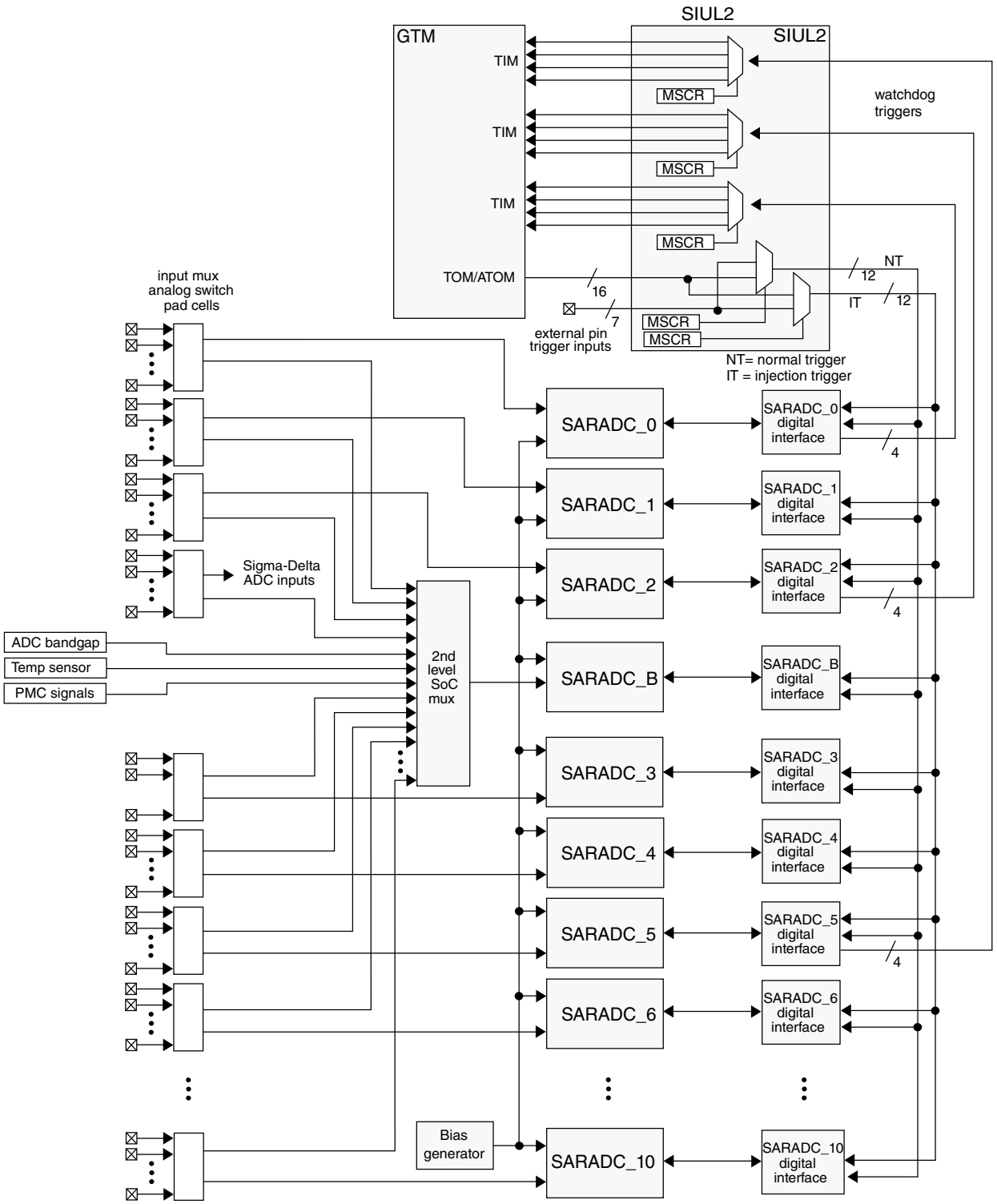


Figure 38-6. SAR ADC integration diagram

38.2.2.2 SAR ADC analog input pin multiplexing

All analog input pins routed to SARADC_B and another fast SAR ADC are multiplexed with a dual analog input switch pad cell. Analog input pins that only connect to SARADC_B use a single analog switch pad cell.

Note

Simultaneous sampling of two ADCs on same analog input is not allowed.

A second level of multiplexing is required for SARADC_B to reduce the parasitic capacitance on the analog input due to the 84:1 input mux. The second level muxing is done in an SoC level IP block so that it can be optimally placed for routing. The analog switches in the second level mux block are controlled by the SAR ADC digital interface block.

The mapping of analog input pins to the SAR converters is given in Table 38-1.

38.2.2.3 SAR ADC clock sources

Each SAR ADC digital interface on MPC5777M has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the ADC clock (SAR_CLK). See Clock generation for more detail on the clock sources.

The input clock for the SAR ADC analog blocks is clocked by the SAR_CLK. The following figure shows a block diagram of the SAR ADC clock architecture. The maximum frequency for the clock input to the converter is 14.6 MHz.

To avoid crosstalk issues, all SAR ADCs run at the same clock frequency. The SAR digital interface is clocked from the PBRIDGE*n*_CLK clock. All SAR analog blocks are clocked from the output of a single divider of the SAR_CLK at the device level.

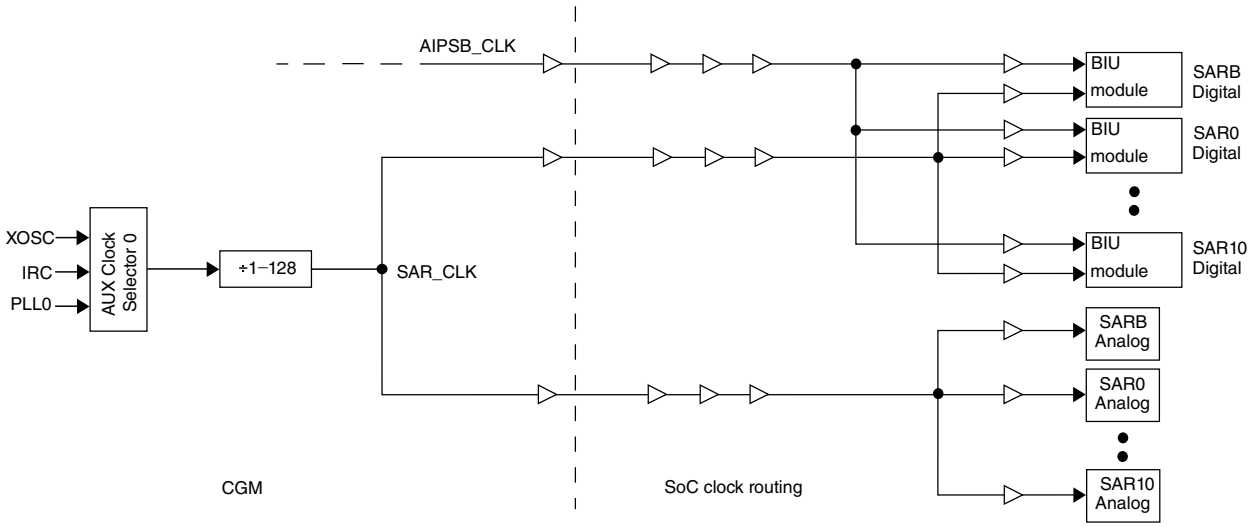


Figure 38-7. SAR ADC clock diagram

38.2.2.4 SAR ADC triggering

Each SAR ADC has 16 independent trigger inputs for starting conversion on an analog input. Selection from the 16 trigger inputs is done inside the SAR ADC digital wrapper. Each SAR ADC on the device has an independent set of trigger inputs, but the same sources are used for each SAR ADC, so only one trigger input is required for each SAR ADC.

The trigger sources for each SAR ADC on the device are either external input pins or GTM channels. Each external input can selectively trigger any of the SAR ADCs on the device. The synchronization to the PBRIDGE clock and digital filtering of the external input pin triggers is done in the system integration logic.

Each trigger input supports software selectable options for rising edge, falling edge, rising and falling edge, or level-based trigger assertion. The trigger selection for each SAR ADC and edge/level selection is located in the SIUL2. The MPC5777M SoC diagram for the SAR ADC trigger inputs is given in the following figure.

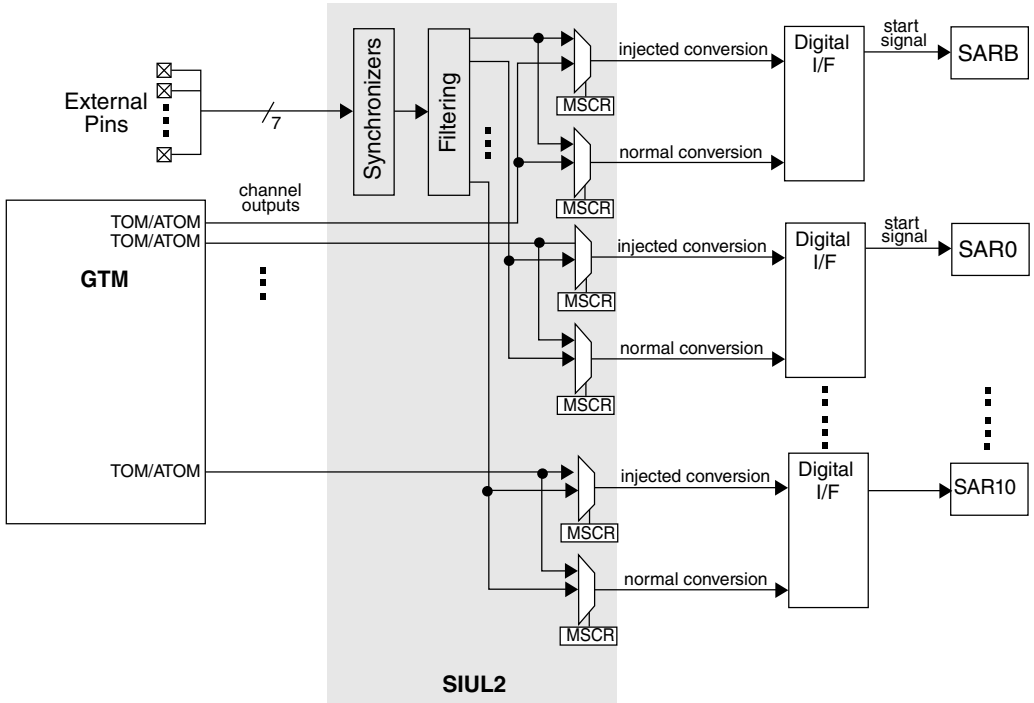


Figure 38-8. SARADC input trigger diagram

Trigger source selection for each SAR ADC is given in the SIUL2 chapter of this document.

38.2.2.5 SAR ADC alternate reference

The SAR ADCs support connection to more than one conversion reference. The alternate references are muxed with analog input pins on MPC5777M. The analog input pins used as alternate references to the respective SAR ADCs are given in the following table.

Table 38-14. SARADC alternate references

SARADC	Alternate reference pin
SARADC_B	AN[8]
SARADC_2	AN[16]

To meet safety requirements, the reference for specific SAR analog inputs cannot be changed. For MPC5777M, these are SARADC_B channels 4–7 and 35, and all other SAR analog inputs. For these SAR ADC instances that do not enable an alternate reference, ICDR*n*[REFSEL] is a read-only bit. There is no option to connect these pins as analog references for the device for safety considerations.

Alternate reference selection is enabled only for the following channels, for which ICDR*n*[REFSEL] is read-write bit:

- SARADC_B channels (0–3, 8–34, 36–95): See Table 38-20, SAR ADC register definitions.
- All SAR2 channels (16–19, 22–23): See Table 38-25, SAR ADC register definitions.

38.2.2.6 SAR absolute voltage reference

MPC5777M provides an internal, absolute voltage reference that can be read by SARADC_B. The ADC conversion channel number for the absolute reference, ADC Bandgap Reference, is given in Table 38-16.

38.2.2.7 SAR ADC diagnostic

38.2.2.7.1 Self Test features

Following are the Self Test features:

- The ADCBIAS provides four different voltage levels defined as GND, $V_{ref}/3$, $2*V_{ref}/3$ and V_{ref}. The self test internal reference voltage precision is provided in the SARn ADC electrical specification table of the device datasheet.
- The impedance requirement has been satisfied putting a physical poly resistor R_{out} = 20 kΩ typical characterized by a process and temperature variation of ±30%.
- The impedance of the switches that are connecting the different references to the ADC input has not been considered in the impedance calculation.
- The impedance of the switches from the reference generation to the ADC input has been matched with the one of the switches that are connecting the device pin to the ADC input.
- The sampling time of SARADC_B needs to be extended to 2.2 ms due to long settling time induced by the high impedance of the source (20 kΩ).
- The SAR ADC Digital Interface provides the necessary control to select the required switches S1–S7 (shown in the following figure) to perform the Self Test.

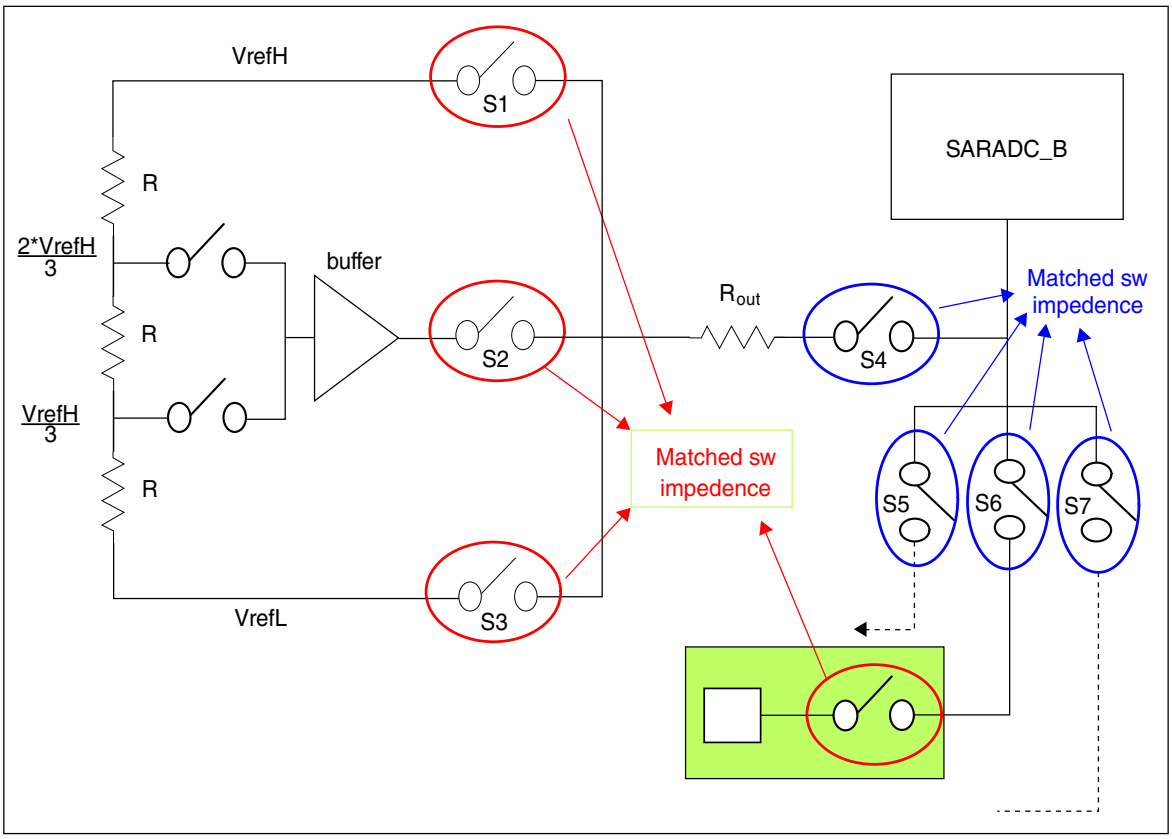


Figure 38-9. Self Test implementation

38.2.2.7.2 Internal reference

The SARB ADC provides the ability to sample and convert four internal voltages through a 20 kΩ source impedance. These voltages are as follows:

- $V_{DD\_HV\_ADR\_S}$
- 1/3 $V_{DD\_HV\_ADR\_S}$
- 2/3 $V_{DD\_HV\_ADR\_S}$
- $V_{SS\_HV\_ADR\_S}$

Conversion of an internal reference voltage works the same as that for a normal conversion, with the exception of the conversion time. Due to the 20 kΩ source impedance, the sampling time is much higher than the normal time.

Enabling the sampling and selection of an internal reference is independent of input channel selection for SARADC_B.

38.2.2.7.3 Analog input pin programmable pullup/pulldown

An open or short circuit condition is detectable within the device for analog input pins. There is a programmable pullup/pulldown pad cell at the input pad. This allows for application testing of the mux input logic. The analog input pin diagram with the pullup/pulldown is shown in the following figure. The pullup/pulldown is included inside the analog input pad cell. The pullup/pulldown is also included in the single switch analog input pad cell.

Control of the pullup/pulldown is independently done in the SIUL2 MSCR registers for each analog input pin.

Only the analog input pins specified in Table 38-1 have the pullup/pulldown feature.

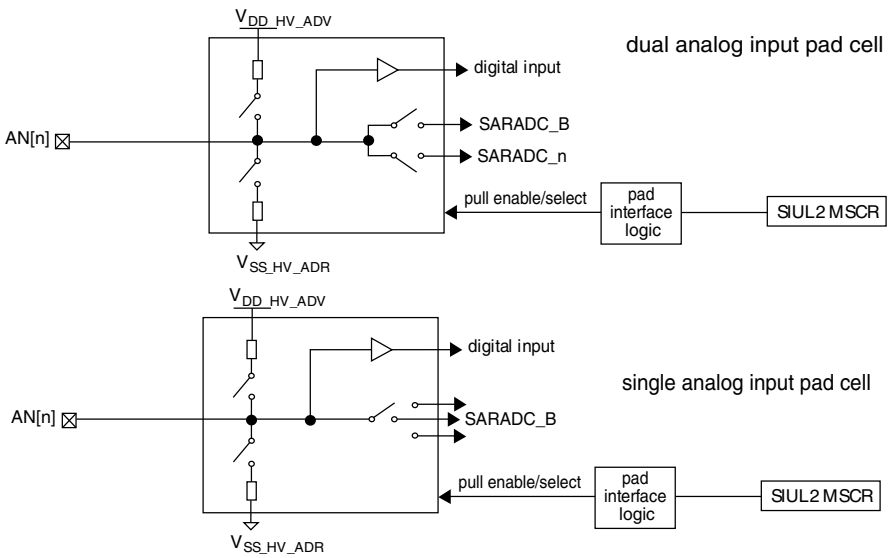


Figure 38-10. Analog input pin pullup/pulldown pad cell

38.2.2.7.4 Analog input pin switch to $V_{SS\_HV\_ADV}$

Two analog input pins on MPC5777M have a programmable switch to $V_{SS\_HV\_ADV}$ to discharge the input. This switch is implemented on analog input pins AN7 and AN35 for MPC5777M. The analog input diagram for AN35 is given in the following figure. The default state of the switch for both inputs is OFF.

Figure 38-11. Analog input AN35 — discharge switch to AVSS

The ground switches for AN7/AN35 are controlled independently from the input switch to the ADC. The ground switch for AN7/AN35 is controlled in input multiplexing MSCR registers (MSCR[WPDE]) in the SIUL2. The MSCR register provides selection between open or closed switch, with open being the default state.

The ground switches can be used to periodically discharge the associated channels.

38.2.2.8 SARADC channel assignment

The input multiplexer channel assignments for the SARADCs is provided in Table 38-15.

38.2.2.8.1 Internal voltage monitor channel assignments

Each internal voltage monitor has a unique channel assignment, and is treated by software like any other A/D conversion. Access to the internal monitor channels is required for SARADC_B only. The internal voltage monitor channel assignments for SARADC_B are given in Table 38-16.

Table 38-15. SAR analog input channel assignment

Analog input pin	SARADC_B input channel	Fast SAR	Fast SAR channel	Description
AN[3:0]	3–0	—	—	Analog input pin
AN[7:4]	7–4	SARADC_0	7–4	Analog input pin, fixed reference, AN[7] gnd switch
AN[15:8]	15–8	SARADC_1	15–8	Analog input pin
AN[19:16]	19–16	SARADC_2	19–16	Analog input pin
AN[21:20]	21–20	—	—	Analog input pin
AN[23:22]	23–22	SARADC_2	23–22	Analog input pin
AN[25:24]	25–24	—	—	Analog input pin
AN[29:26]	29–26	SARADC_3	29–26	Analog input pin
AN[31:30]	31–30	SARADC_3	31–30	Analog input pin
AN[35:32]	35–32	SARADC_4	35–32	Analog input pin, AN[35] fixed reference and gnd switch
AN[39:36]	39–36	—	—	Analog input pin
AN[43:40]	43–40	SARADC_5	43–40	Analog input pin
AN[47:44]	47–44	—	—	Analog input pin
AN[51:48]	51–48	SARADC_6	51–48	Analog input pin
AN[53:52]	53–52	—	—	Analog input pin
AN[59:56]	59–56	SARADC_7	59–56	Analog input pin
AN[61:60]	61–60	—	—	Analog input pin
AN[63:62]	63–62	—	—	Analog input pin
AN[67:64]	67–64	SARADC_8	67–64	Analog input pin

Table continues on the next page...

Table 38-15. SAR analog input channel assignment (continued)

Analog input pin	SARADC_B input channel	Fast SAR	Fast SAR channel	Description
AN[71:68]	71–68	—	—	Analog input pin
AN[75:72]	75–72	SARADC_9	75–72	Analog input pin
AN[79:76]	79–76	—	—	Analog input pin
AN[83:80]	83–80	SARADC_10	83–80	Analog input pin

Table 38-16. SARADC_B analog test channel assignment

SARADC_B input channel	Fast SAR	Fast SAR channel	Description
96	—	—	VDD_HV_PMC
97	—	—	VDD_HV_ADV_S
98	—	—	VDD_HV_JTAG_OSC
99	—	—	VDD_HV_FLA
100	—	—	VDD_HV_IO_MAIN
101	—	—	VDD_HV_IO_FLEX
102	—	—	Reserved
103	—	—	VDD_HV_IO_FLEXE
104	—	—	VDD_LV
105	—	—	VDD_HV_IO_EBI
106	—	—	Unbuffered PMC Trim Bandgap Reference (used in PMC)
107	—	—	2nd bandgap (Trimmed), (ADC mode=60, ADC register=35 from power management)
108	—	—	Buffered PMC Trim Bandgap Reference (used in Pad)
109	—	—	VSS_HV_ADV (Ground Supply for ADC)
110	—	—	PMC generic input 0
111	—	—	Reserved
112[1]	—	—	HV regulator supply LVD falling edge 280_C
113[1]	—	—	HV regulator supply HVD rising edge 600_C
114[1]	—	—	HV flash supply LVD falling edge 280_F
115[1]	—	—	HV flash supply HVD rising edge 360_F
116[1]	—	—	LV core supply LVD falling edge 114_C
117[1]	—	—	LV core supply HVD falling edge 140_C
118[1]	—	—	LV flash supply LVD falling edge 114_F
119	—	—	Reserved
120	—	—	Temperature sensor
121	—	—	ADC bandgap reference
122	—	—	Reserved[2]
123	—	—	Reserved[2]
124	—	—	SAR BIAS 0 — VSS_HV_ADR_S through 20 KΩ source impedance

Table continues on the next page...

Table 38-16. SARADC_B analog test channel assignment (continued)

SARADC_B input channel	Fast SAR	Fast SAR channel	Description
125	—	—	SAR BIAS 1 — 1/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance
126	—	—	SAR BIAS 2 — 2/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance
127	—	—	SAR BIAS 3 — (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance

1. When doing conversions on this channel, it is recommended to mask the corresponding HVD/LVD from generating a reset for the duration of the conversion when this channel is enabled.
2. Channels 122 and 123 are reserved for factory use only.

38.2.2.8.2 SARADC_B interface to PMC

The connection of the PMC to SARADC_B ADC is shown in the following figure. SARADC_B ADC channel 110 connects to the PMC. The PMC contains an analog mux that passes internal voltage nodes in addition to those given in Table 38-16. The PMC mux is programmed in a register in the PMC digital interface. To convert an internal PMC voltage, the user programs the PMC analog mux selection, and then programs SARADC_B to convert PMC channel 110. See the Power Management chapter of this document for more info on additional internal voltages that can be read by the SARADC_B ADC.

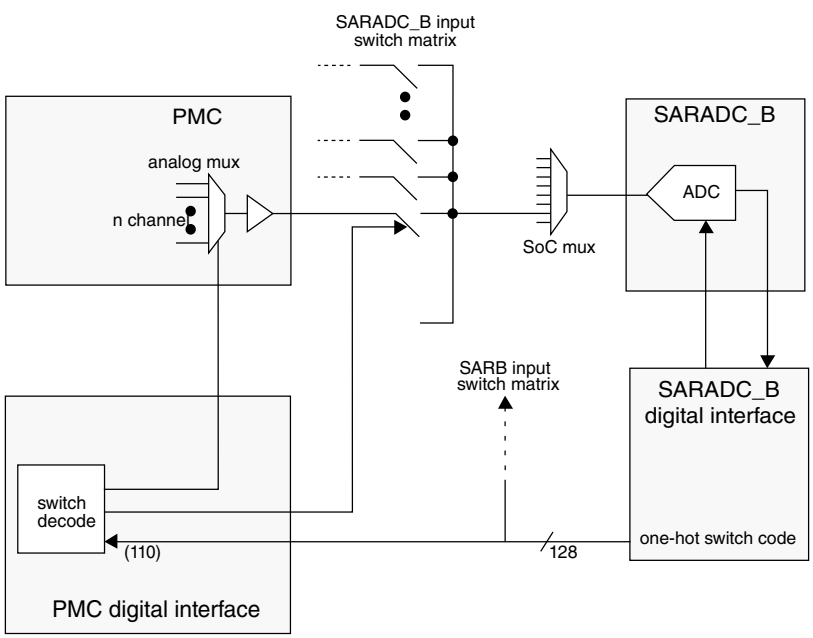


Figure 38-12. SARADC_B ADC read of internal PMC voltages

38.2.2.9 SAR ADC register definitions

Each of the following sections shows register definitions for one SAR ADC instance:

- SARADC_B register definitions
- SARADC_0 register definitions
- SARADC_1 register definitions
- SARADC_2 register definitions
- SARADC_3 register definitions
- SARADC_4 register definitions
- SARADC_5 register definitions
- SARADC_6 register definitions
- SARADC_7 register definitions
- SARADC_8 register definitions
- SARADC_9 register definitions
- SARADC_10 register definitions

The following tables provide a key for the register definition tables. Test channels and external channels are only used for SARADC_B.

Table 38-17. SAR ADC register descriptions

Name	Description	User Access
MCR	Main Configuration Register	read/write
MSR	Main Status Register	read
ISR	Interrupt Status Register	read
ICIPR*n*	Internal Channel Interrupt Pending Registers	read
IMR	Interrupt Mask Register	read/write
ICIMR*n*	Internal Channel Interrupt Mask Registers	read/write
WTISR	Watchdog Threshold Interrupt Status Register	read
WTIMR	Watchdog Threshold Interrupt Mask Register	read/write
DMAE	DMA Enable Register	read/write
ICDSR*n*	Internal Channel DMA Select Registers	read/write
WTHRHLR0–3	Watchdog Threshold Registers 0–3	read/write
CTR*n*	Conversion Timing Registers 0–3	read/write

Table continues on the next page...

Table 38-17. SAR ADC register descriptions (continued)

Name	Description	User Access
ICNCMR*n*	Internal Channel Normal Conversion Mask Registers	read/write
ICJCMR*n*	Internal Channel Injected Conversion Mask Registers	read/write
PDEDR	Power Down Exit Delay Register	read/write
ICDR*n*	Internal Channel Data Registers 0–95	mixed
WTHRHLR4–15	Watchdog Threshold Registers 4–15	read/write
ICWSELR*n*	Internal Channel Watchdog Select Registers	read/write
ICWENR*n*	Internal Channel Watchdog Enable Registers	read/write
ICAWORR*n*	Internal Channel Analog Watchdog Out of Range Registers	read

Table 38-18. SAR ADC test channel register descriptions (SARADC_B only)

Name	Description	User Access
TCIPR	Test Channel Interrupt Pending Register	read
TCIMR	Test Channel Interrupt Mask Register	read/write
TCDSR	Test Channel DMA Select Register	read/write
TCNCMR	Test Channel Normal Conversion Mask Register	read/write
TCJCMR	Test Channel Injected Conversion Mask Register	read/write
TCWSELR*n*	Test Channel Watchdog Select Register	read/write
TCWENR	Test Channel Watchdog Enable Register	read/write
TCAWORR	Test Channel Analog Watchdog Out of Range Register	read
TCCAPR*n*	Test Channel Connection with Analog Pin Registers	read/write
TCDR*n*	Test Channel Data Register 96-127	read/write

Table 38-19. SAR ADC external channel register descriptions (SARADC_B only)

Name	Description	User Access
ECDSDR	External Channel Decode Signals Delay Register	read/write
ECIPR*n*	External Channel Interrupt Pending Registers	read
ECIMR*n*	External Channel Interrupt Mask Registers	read/write
ECDSR*n*	External Channel DMA Select Registers	read/write
ECNCMR*n*	External Channel Normal Conversion Mask Registers	read/write
ECJCMR*n*	External Channel Injected Conversion Mask Registers	read/write
ECWSELR*n*	External Channel Watchdog Select Registers	read/write
ECWENR*n*	External Channel Watchdog Enable Registers	read/write
ECAWORR*n*	External Channel Analog Watchdog Out of Range registers	read
ECMICR*n*	External Channel Mapping to Internal Channel Registers	read/write
ECDR128-211	External Channel Data Registers 128-211	read/write

38.2.2.9.1 SARADC_B register definitions

Table 38-20. SARADC_B register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADCB_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADCB_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADCB_ICIPR0	R	EOC_CH[31]	EOC_CH[30]	EOC_CH[29]	EOC_CH[28]	EOC_CH[27]	EOC_CH[26]	EOC_CH[25]	EOC_CH[24]	EOC_CH[23]	EOC_CH[22]	EOC_CH[21]	EOC_CH[20]	EOC_CH[19]	EOC_CH[18]	EOC_CH[17]	EOC_CH[16]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[15]	EOC_CH[14]	EOC_CH[13]	EOC_CH[12]	EOC_CH[11]	EOC_CH[10]	EOC_CH[9]	EOC_CH[8]	EOC_CH[7]	EOC_CH[6]	EOC_CH[5]	EOC_CH[4]	EOC_CH[3]	EOC_CH[2]	EOC_CH[1]	EOC_CH[0]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ICIPR1	R	0	0	EOC_CH[61]	EOC_CH[60]	EOC_CH[59]	EOC_CH[58]	EOC_CH[57]	EOC_CH[56]	0	0	EOC_CH[53]	EOC_CH[52]	EOC_CH[51]	EOC_CH[50]	EOC_CH[49]	EOC_CH[48]
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[47]	EOC_CH[46]	EOC_CH[45]	EOC_CH[44]	EOC_CH[43]	EOC_CH[42]	EOC_CH[41]	EOC_CH[40]	EOC_CH[39]	EOC_CH[38]	EOC_CH[37]	EOC_CH[36]	EOC_CH[35]	EOC_CH[34]	EOC_CH[33]	EOC_CH[32]

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ICIPR2	R	0	0	0	0	0	0	0	0	EOC_CH[87]	EOC_CH[86]	EOC_CH[85]	EOC_CH[84]	EOC_CH[83]	EOC_CH[82]	EOC_CH[81]	EOC_CH[80]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[79]	EOC_CH[78]	EOC_CH[77]	EOC_CH[76]	EOC_CH[75]	EOC_CH[74]	EOC_CH[73]	EOC_CH[72]	EOC_CH[71]	EOC_CH[70]	EOC_CH[69	EOC_CH[68]	EOC_CH[67]	EOC_CH[66]	EOC_CH[65]	EOC_CH[64]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADCB_ICIMR0	R	IM_CH[31]	IM_CH[30]	IM_CH[29]	IM_CH[28]	IM_CH[27]	IM_CH[26]	IM_CH[25]	IM_CH[24]	IM_CH[23]	IM_CH[22]	IM_CH[21]	IM_CH[20]	IM_CH[19]	IM_CH[18]	IM_CH[17]	IM_CH[16]
	W																
	R	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	IM_CH[3]	IM_CH[2]	IM_CH[1]	IM_CH[0]
	W																
SARADCB_ICIMR1	R	0	0	IM_CH[61]	IM_CH[60]	IM_CH[59]	IM_CH[58]	IM_CH[57]	IM_CH[56]	0	0	IM_CH[53]	IM_CH[52]	IM_CH[51]	IM_CH[50]	IM_CH[49]	IM_CH[48]
	W																
	R	IM_CH[47]	IM_CH[46]	IM_CH[45]	IM_CH[44]	IM_CH[43]	IM_CH[42]	IM_CH[41]	IM_CH[40]	IM_CH[39]	IM_CH[38]	IM_CH[37]	IM_CH[36]	IM_CH[35]	IM_CH[34]	IM_CH[33]	IM_CH[32]
	W																
SARADCB_ICIMR2	R	0	0	0	0	0	0	0	0	IM_CH[87]	IM_CH[86]	IM_CH[85]	IM_CH[84]	IM_CH[83]	IM_CH[82]	IM_CH[81]	IM_CH[80]
	W																
	R	IM_CH[79]	IM_CH[78]	IM_CH[77]	IM_CH[76]	IM_CH[75]	IM_CH[74]	IM_CH[73]	IM_CH[72]	IM_CH[71]	IM_CH[70]	IM_CH[69]	IM_CH[68]	IM_CH[67]	IM_CH[66]	IM_CH[65]	IM_CH[64]
	W																

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADCB_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADCB_ICDSR0	R	DS_CH[31]	DS_CH[30]	DS_CH[29]	DS_CH[28]	DS_CH[27]	DS_CH[26]	DS_CH[25]	DS_CH[24]	DS_CH[23]	DS_CH[22]	DS_CH[21]	DS_CH[20]	DS_CH[19]	DS_CH[18]	DS_CH[17]	DS_CH[16]
	W																
	R	DS_CH[15]	DS_CH[14]	DS_CH[13]	DS_CH[12]	DS_CH[11]	DS_CH[10]	DS_CH[9]	DS_CH[8]	DS_CH[7]	DS_CH[6]	DS_CH[5]	DS_CH[4]	DS_CH[3]	DS_CH[2]	DS_CH[1]	DS_CH[0]
	W																
SARADCB_ICDSR1	R	0	0	DS_CH[61]	DS_CH[60]	DS_CH[59]	DS_CH[58]	DS_CH[57]	DS_CH[56]	0	0	DS_CH[53]	DS_CH[52]	DS_CH[51]	DS_CH[50]	DS_CH[49]	DS_CH[48]
	W																
	R	DS_CH[47]	DS_CH[46]	DS_CH[45]	DS_CH[44]	DS_CH[43]	DS_CH[42]	DS_CH[41]	DS_CH[40]	DS_CH[39]	DS_CH[38]	DS_CH[37]	DS_CH[36]	DS_CH[35]	DS_CH[34]	DS_CH[33]	DS_CH[32]
	W																
SARADCB_ICDSR2	R	0	0	0	0	0	0	0	0	DS_CH[87]	DS_CH[86]	DS_CH[85]	DS_CH[84]	DS_CH[83]	DS_CH[82]	DS_CH[81]	DS_CH[80]
	W																
	R	DS_CH[79]	DS_CH[78]	DS_CH[77]	DS_CH[76]	DS_CH[75]	DS_CH[74]	DS_CH[73]	DS_CH[72]	DS_CH[71]	DS_CH[70]	DS_CH[69]	DS_CH[68]	DS_CH[67]	DS_CH[66]	DS_CH[65]	DS_CH[64]
	W																

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_WTHRHLR0–WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADCB_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADCB_ICNCMR0	R																
	W	NCE_CH[31]	NCE_CH[30]	NCE_CH[29]	NCE_CH[28]	NCE_CH[27]	NCE_CH[26]	NCE_CH[25]	NCE_CH[24]	NCE_CH[23]	NCE_CH[22]	NCE_CH[21]	NCE_CH[20]	NCE_CH[19]	NCE_CH[18]	NCE_CH[17]	NCE_CH[16]
	R																
	W	NCE_CH[15]	NCE_CH[14]	NCE_CH[13]	NCE_CH[12]	NCE_CH[11]	NCE_CH[10]	NCE_CH[9]	NCE_CH[8]	NCE_CH[7]	NCE_CH[6]	NCE_CH[5]	NCE_CH[4]	NCE_CH[3]	NCE_CH[2]	NCE_CH[1]	NCE_CH[0]
SARADCB_ICNCMR1	R	0	0							0	0						
	W			NCE_CH[61]	NCE_CH[60]	NCE_CH[59]	NCE_CH[58]	NCE_CH[57]	NCE_CH[56]			NCE_CH[53]	NCE_CH[52]	NCE_CH[51]	NCE_CH[50]	NCE_CH[49]	NCE_CH[48]
	R																
	W	NCE_CH[47]	NCE_CH[46]	NCE_CH[45]	NCE_CH[44]	NCE_CH[43]	NCE_CH[42]	NCE_CH[41]	NCE_CH[40]	NCE_CH[39]	NCE_CH[38]	NCE_CH[37]	NCE_CH[36]	NCE_CH[35]	NCE_CH[34]	NCE_CH[33]	NCE_CH[32]
SARADCB_ICNCMR2	R	0	0	0	0	0	0	0	0								
	W									NCE_CH[87]	NCE_CH[86]	NCE_CH[85]	NCE_CH[84]	NCE_CH[83]	NCE_CH[82]	NCE_CH[81]	NCE_CH[80]
	R																
	W	NCE_CH[79]	NCE_CH[78]	NCE_CH[77]	NCE_CH[76]	NCE_CH[75]	NCE_CH[74]	NCE_CH[73]	NCE_CH[72]	NCE_CH[71]	NCE_CH[70]	NCE_CH[69]	NCE_CH[68]	NCE_CH[67]	NCE_CH[66]	NCE_CH[65]	NCE_CH[64]
SARADCB_ICJCMR0	R																
	W	JCE_CH[31]	JCE_CH[30]	JCE_CH[29]	JCE_CH[28]	JCE_CH[27]	JCE_CH[26]	JCE_CH[25]	JCE_CH[24]	JCE_CH[23]	JCE_CH[22]	JCE_CH[21]	JCE_CH[20]	JCE_CH[19]	JCE_CH[18]	JCE_CH[17]	JCE_CH[16]

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	JCE_CH[15]	JCE_CH[14]	JCE_CH[13]	JCE_CH[12]	JCE_CH[11]	JCE_CH[10]	JCE_CH[9]	JCE_CH[8]	JCE_CH[7]	JCE_CH[6]	JCE_CH[5]	JCE_CH[4]	JCE_CH[3]	JCE_CH[2]	JCE_CH[1]	JCE_CH[0]
	W																
SARADCB_ICJCMR1	R	0	0	JCE_CH[61]	JCE_CH[60]	JCE_CH[59]	JCE_CH[58]	JCE_CH[57]	JCE_CH[56]	0	0	JCE_CH[53]	JCE_CH[52]	JCE_CH[51]	JCE_CH[50]	JCE_CH[49]	JCE_CH[48]
	W																
	R	JCE_CH[47]	JCE_CH[46]	JCE_CH[45]	JCE_CH[44]	JCE_CH[43]	JCE_CH[42]	JCE_CH[41]	JCE_CH[40]	JCE_CH[39]	JCE_CH[38]	JCE_CH[37]	JCE_CH[36]	JCE_CH[35]	JCE_CH[34]	JCE_CH[33]	JCE_CH[32]
	W																
SARADCB_ICJCMR2	R	0	0	0	0	0	0	0	0	JCE_CH[87]	JCE_CH[86]	JCE_CH[85]	JCE_CH[84]	JCE_CH[83]	JCE_CH[82]	JCE_CH[81]	JCE_CH[80]
	W																
	R	JCE_CH[79]	JCE_CH[78]	JCE_CH[77]	JCE_CH[76]	JCE_CH[75]	JCE_CH[74]	JCE_CH[73]	JCE_CH[72]	JCE_CH[71]	JCE_CH[70]	JCE_CH[69]	JCE_CH[68]	JCE_CH[67]	JCE_CH[66]	JCE_CH[65]	JCE_CH[64]
	W																
SARADCB_ICDR0-3, ICDR8-34, ICDR36-95	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADCB_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADCB_ICDR4-7, ICDR35	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADCB_WTHRHLR4–WTHRHLR7	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ICWSELR0	R	0	WSEL_CH7			0	WSEL_CH6			0	WSEL_CH5			0	WSEL_CH4		
	W																
	R	0	WSEL_CH3			0	WSEL_CH2			0	WSEL_CH1			0	WSEL_CH0		
	W																
SARADCB_ICWSELR1	R	0	WSEL_CH15			0	WSEL_CH14			0	WSEL_CH13			0	WSEL_CH12		
	W																
	R	0	WSEL_CH11			0	WSEL_CH10			0	WSEL_CH9			0	WSEL_CH8		
	W																
SARADCB_ICWSELR2	R	0	WSEL_CH23			0	WSEL_CH22			0	WSEL_CH21			0	WSEL_CH20		
	W																
	R	0	WSEL_CH19			0	WSEL_CH18			0	WSEL_CH17			0	WSEL_CH16		
	W																
SARADCB_ICWSELR3	R	0	WSEL_CH31			0	WSEL_CH30			0	WSEL_CH29			0	WSEL_CH28		
	W																
	R	0	WSEL_CH27			0	WSEL_CH26			0	WSEL_CH25			0	WSEL_CH24		
	W																
SARADCB_ICWSELR4	R	0	WSEL_CH39			0	WSEL_CH38			0	WSEL_CH37			0	WSEL_CH36		
	W																
	R	0	WSEL_CH35			0	WSEL_CH34			0	WSEL_CH33			0	WSEL_CH32		
	W																
SARADCB_ICWSELR5	R	0	WSEL_CH47			0	WSEL_CH46			0	WSEL_CH45			0	WSEL_CH44		
	W																
	R	0	WSEL_CH43			0	WSEL_CH42			0	WSEL_CH41			0	WSEL_CH40		
	W																
SARADCB_ICWSELR6	R	0	0	0	0	0	0	0	0	0	WSEL_CH53			0	WSEL_CH52		
	W																
	R	0	WSEL_CH51			0	WSEL_CH50			0	WSEL_CH49			0	WSEL_CH48		
	W																
SARADCB_ICWSELR7	R	0	0	0	0	0	0	0	0	0	WSEL_CH61			0	WSEL_CH60		
	W																
	R	0	WSEL_CH59			0	WSEL_CH58			0	WSEL_CH57			0	WSEL_CH56		
	W																
SARADCB_ICWSELR8	R	0	WSEL_CH71			0	WSEL_CH70			0	WSEL_CH69			0	WSEL_CH68		
	W																
	R	0	WSEL_CH67			0	WSEL_CH66			0	WSEL_CH65			0	WSEL_CH64		
	W																
SARADCB_ICWSELR9	R	0	WSEL_CH79			0	WSEL_CH78			0	WSEL_CH77			0	WSEL_CH76		

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	0	WSEL_CH75			0	WSEL_CH74			0	WSEL_CH73			0	WSEL_CH72		
	W																
SARADCB_ICWSELR10	R	0	WSEL_CH87			0	WSEL_CH86			0	WSEL_CH85			0	WSEL_CH84		
	W																
	R	0	WSEL_CH83			0	WSEL_CH82			0	WSEL_CH81			0	WSEL_CH80		
	W																
SARADCB_ICWENR0	R	WEN_CH[31]	WEN_CH[30]	WEN_CH[29]	WEN_CH[28]	WEN_CH[27]	WEN_CH[26]	WEN_CH[25]	WEN_CH[24]	WEN_CH[23]	WEN_CH[22]	WEN_CH[21]	WEN_CH[20]	WEN_CH[19]	WEN_CH[18]	WEN_CH[17]	WEN_CH[16]
	W																
	R	WEN_CH[15]	WEN_CH[14]	WEN_CH[13]	WEN_CH[12]	WEN_CH[11]	WEN_CH[10]	WEN_CH[9]	WEN_CH[8]	WEN_CH[7]	WEN_CH[6]	WEN_CH[5]	WEN_CH[4]	WEN_CH[3]	WEN_CH[2]	WEN_CH[1]	WEN_CH[0]
	W																
SARADCB_ICWENR1	R	0	0	WEN_CH[61]	WEN_CH[60]	WEN_CH[59]	WEN_CH[58]	WEN_CH[57]	WEN_CH[56]	0	0	WEN_CH[53]	WEN_CH[52]	WEN_CH[51]	WEN_CH[50]	WEN_CH[49]	WEN_CH[48]
	W																
	R	WEN_CH[47]	WEN_CH[46]	WEN_CH[45]	WEN_CH[44]	WEN_CH[43]	WEN_CH[42]	WEN_CH[41]	WEN_CH[40]	WEN_CH[39]	WEN_CH[38]	WEN_CH[37]	WEN_CH[36]	WEN_CH[35]	WEN_CH[34]	WEN_CH[33]	WEN_CH[32]
	W																
SARADCB_ICWENR2	R	0	0	0	0	0	0	0	0	WEN_CH[87]	WEN_CH[86]	WEN_CH[85]	WEN_CH[84]	WEN_CH[83]	WEN_CH[82]	WEN_CH[81]	WEN_CH[80]
	W																
	R	WEN_CH[79]	WEN_CH[78]	WEN_CH[77]	WEN_CH[76]	WEN_CH[75]	WEN_CH[74]	WEN_CH[73]	WEN_CH[72]	WEN_CH[71]	WEN_CH[70]	WEN_CH[69]	WEN_CH[68]	WEN_CH[67]	WEN_CH[66]	WEN_CH[65]	WEN_CH[64]
	W																
SARADCB_ICAWORR0	R	AWOR_CH[31]	AWOR_CH[30]	AWOR_CH[29]	AWOR_CH[28]	AWOR_CH[27]	AWOR_CH[26]	AWOR_CH[25]	AWOR_CH[24]	AWOR_CH[23]	AWOR_CH[22]	AWOR_CH[21]	AWOR_CH[20]	AWOR_CH[19]	AWOR_CH[18]	AWOR_CH[17]	AWOR_CH[16]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-20. SARADC_B register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	AWOR_CH[15]	AWOR_CH[14]	AWOR_CH[13]	AWOR_CH[12]	AWOR_CH[11]	AWOR_CH[10]	AWOR_CH[9]	AWOR_CH[8]	AWOR_CH[7]	AWOR_CH[6]	AWOR_CH[5]	AWOR_CH[4]	AWOR_CH[3]	AWOR_CH[2]	AWOR_CH[1]	AWOR_CH[0]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ICAWORR1	R	0	0	AWOR_CH[61]	AWOR_CH[60]	AWOR_CH[59]	AWOR_CH[58]	AWOR_CH[57]	AWOR_CH[56]	0	0	AWOR_CH[53]	AWOR_CH[52]	AWOR_CH[51]	AWOR_CH[50]	AWOR_CH[49]	AWOR_CH[48]
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH[47]	AWOR_CH[46]	AWOR_CH[45]	AWOR_CH[44]	AWOR_CH[43]	AWOR_CH[42]	AWOR_CH[41]	AWOR_CH[40]	AWOR_CH[39]	AWOR_CH[38]	AWOR_CH[37]	AWOR_CH[36]	AWOR_CH[35]	AWOR_CH[34]	AWOR_CH[33]	AWOR_CH[32]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ICAWORR0	R	0	0	0	0	0	0	0	0	AWOR_CH[87]	AWOR_CH[86]	AWOR_CH[85]	AWOR_CH[84]	AWOR_CH[83]	AWOR_CH[82]	AWOR_CH[81]	AWOR_CH[80]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH[79]	AWOR_CH[78]	AWOR_CH[77]	AWOR_CH[76]	AWOR_CH[75]	AWOR_CH[74]	AWOR_CH[73]	AWOR_CH[72]	AWOR_CH[71]	AWOR_CH[70]	AWOR_CH[69]	AWOR_CH[68]	AWOR_CH[67]	AWOR_CH[66]	AWOR_CH[65]	AWOR_CH[64]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 38-21. SARADC_B test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_TCIPR	R	EOC_CH[127]	EOC_CH[126]	EOC_CH[125]	EOC_CH[124]	0	EOC_CH[122]	EOC_CH[121]	EOC_CH[120]	0	EOC_CH[118]	EOC_CH[117]	EOC_CH[116]	EOC_CH[115]	EOC_CH[114]	EOC_CH[113]	EOC_CH[112]

Table continues on the next page...

Table 38-21. SARADC_B test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W	w1c	w1c	w1c	w1c		w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	0	EOC_CH[110]	0	EOC_CH[108]	0	EOC_CH[106]	0	EOC_CH[104]	0	0	EOC_CH[101]	EOC_CH[100]	EOC_CH[99]	EOC_CH[98]	EOC_CH[97]	EOC_CH[96]
	W		w1c		w1c		w1c		w1c			w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_TCIMR	R									0							
	W	IM_CH[127]	IM_CH[126]	IM_CH[125]	IM_CH[124]	IM_CH[123]	IM_CH[122]	IM_CH[121]	IM_CH[120]		IM_CH[118]	IM_CH[117]	IM_CH[116]	IM_CH[115]	IM_CH[114]	IM_CH[113]	IM_CH[112]
	R	0		0		0		0		0	0						
	W		IM_CH[110]		IM_CH[108]		IM_CH[106]		IM_CH[104]			IM_CH[101]	IM_CH[100]	IM_CH[99]	IM_CH[98]	IM_CH[97]	IM_CH[96]
SARADCB_TCDSR	R									0							
	W	DS_CH[127]	DS_CH[126]	DS_CH[125]	DS_CH[124]	DS_CH[123]	DS_CH[122]	DS_CH[121]	DS_CH[120]		DS_CH[118]	DS_CH[117]	DS_CH[116]	DS_CH[115]	DS_CH[114]	DS_CH[113]	DS_CH[112]
	R	0		0		0		0		0	0						
	W		DS_CH[110]		DS_CH[108]		DS_CH[106]		DS_CH[104]			DS_CH[101]	DS_CH[100]	DS_CH[99]	DS_CH[98]	DS_CH[97]	DS_CH[96]
SARADCB_TCNCMR	R									0							
	W	NCE_CH[127]	NCE_CH[126]	NCE_CH[125]	NCE_CH[124]	NCE_CH[123]	NCE_CH[122]	NCE_CH[121]	NCE_CH[120]		NCE_CH[118]	NCE_CH[117]	NCE_CH[116]	NCE_CH[115]	NCE_CH[114]	NCE_CH[113]	NCE_CH[112]
	R	0		0		0		0		0	0						
	W		NCE_CH[110]		NCE_CH[108]		NCE_CH[106]		NCE_CH[104]			NCE_CH[101]	NCE_CH[100]	NCE_CH[99]	NCE_CH[98]	NCE_CH[97]	NCE_CH[96]
SARADCB_TCJCMR	R									0							
	W	JCE_CH[127]	JCE_CH[126]	JCE_CH[125]	JCE_CH[124]	JCE_CH[123]	JCE_CH[122]	JCE_CH[121]	JCE_CH[120]		JCE_CH[118]	JCE_CH[117]	JCE_CH[116]	JCE_CH[115]	JCE_CH[114]	JCE_CH[113]	JCE_CH[112]

Table continues on the next page...

Table 38-21. SARADC_B test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	JCE_CH[110]	0	JCE_CH[108]	0	JCE_CH[106]	0	JCE_CH[104]	0	0	JCE_CH[101]	JCE_CH[100]	JCE_CH[99]	JCE_CH[98]	JCE_CH[97]	JCE_CH[96]
	W																
SARADCB_TCWSELR0	R	0	0	0	0	0	0	0	0	0	WSEL_CH101			0	WSEL_CH100		
	W																
	R	0	WSEL_CH99			0	WSEL_CH98			0	WSEL_CH97			0	WSEL_CH96		
	W																
SARADCB_TCWSELR1	R	0	0	0	0	0	WSEL_CH110			0	0	0	0	0	WSEL_CH108		
	W																
	R	0	0	0	0	0	WSEL_CH106			0	0	0	0	0	WSEL_CH104		
	W																
SARADCB_TCWSELR2	R	0	0	0	0	0	WSEL_CH118			0	WSEL_CH117			0	WSEL_CH116		
	W																
	R	0	WSEL_CH115			0	WSEL_CH114			0	WSEL_CH113			0	WSEL_CH112		
	W																
SARADCB_TCWSELR3	R	0	WSEL_CH127			0	WSEL_CH126			0	WSEL_CH125			0	WSEL_CH124		
	W																
	R	0	0	0	0	0	WSEL_CH122			0	WSEL_CH121			0	WSEL_CH120		
	W																
SARADCB_TCWENR	R	WEN_CH[127]	WEN_CH[126]	WEN_CH[125]	WEN_CH[124]	WEN_CH[123]	WEN_CH[122]	WEN_CH[121]	WEN_CH[120]	0	WEN_CH[118]	WEN_CH[117]	WEN_CH[116]	WEN_CH[115]	WEN_CH[114]	WEN_CH[113]	WEN_CH[112]
	W																
	R	0	WEN_CH[110]	0	WEN_CH[108]	0	WEN_CH[106]	0	WEN_CH[104]	0	0	WEN_CH[101]	WEN_CH[100]	WEN_CH[99]	WEN_CH[98]	WEN_CH[97]	WEN_CH[96]
	W																
SARADCB_TCAWORR	R	AWOR_CH[127]	AWOR_CH[126]	AWOR_CH[125]	AWOR_CH[124]	0	AWOR_CH[122]	AWOR_CH[121]	AWOR_CH[120]	0	AWOR_CH[118]	AWOR_CH[117]	AWOR_CH[116]	AWOR_CH[115]	AWOR_CH[114]	AWOR_CH[113]	AWOR_CH[112]
	W	w1c	w1c	w1c	w1c			w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-21. SARADC_B test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	AWOR_CH[110]	0	AWOR_CH[108]	0	AWOR_CH[106]	0	AWOR_CH[104]	0	0	AWOR_CH[101]	AWOR_CH[100]	AWOR_CH_[99]	AWOR_CH_[98]	AWOR_CH_[97]	AWOR_CH_[96]
	W		w1c		w1c		w1c		w1c			w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
SARADCB_TCDR96-101TCDR104, TCDR106, TCDR108, TCDR110, TCDR112-122, TCDR124-127	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

Table 38-22. SARADC_B external channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC_ECDSDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	DSD											
	W																
SARADCB_ECIPR0	R	EOC_CH[159]	EOC_CH[158]	EOC_CH[157]	EOC_CH[156]	EOC_CH[155]	EOC_CH[154]	EOC_CH[153]	EOC_CH[152]	EOC_CH[151]	EOC_CH[150]	EOC_CH[149]	EOC_CH[148]	EOC_CH[147]	EOC_CH[146]	EOC_CH[145]	EOC_CH[144]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-22. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	EOC_CH[143]	EOC_CH[142]	EOC_CH[141]	EOC_CH[140]	EOC_CH[139]	EOC_CH[138]	EOC_CH[137]	EOC_CH[136]	EOC_CH[135]	EOC_CH[134]	EOC_CH[133]	EOC_CH[132]	EOC_CH[131]	EOC_CH[130]	EOC_CH[129]	EOC_CH[128]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECIPR1	R	EOC_CH[191]	EOC_CH[190]	EOC_CH[189]	EOC_CH[188]	EOC_CH[187]	EOC_CH[186]	EOC_CH[185]	EOC_CH[184]	EOC_CH[183]	EOC_CH[182]	EOC_CH[181]	EOC_CH[180]	EOC_CH[179]	EOC_CH[178]	EOC_CH[177]	EOC_CH[176]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOC_CH[175]	EOC_CH[174]	EOC_CH[173]	EOC_CH[172]	EOC_CH[171]	EOC_CH[170]	EOC_CH[169]	EOC_CH[168]	EOC_CH[167]	EOC_CH[166]	EOC_CH[165]	EOC_CH[164]	EOC_CH[163]	EOC_CH[162]	EOC_CH[161]	EOC_CH[160]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECIMR0	R	IM_CH[159]	IM_CH[158]	IM_CH[157]	IM_CH[156]	IM_CH[155]	IM_CH[154]	IM_CH[153]	IM_CH[152]	IM_CH[151]	IM_CH[150]	IM_CH[149]	IM_CH[148]	IM_CH[147]	IM_CH[146]	IM_CH[145]	IM_CH[144]
	W																
	R	IM_CH[143]	IM_CH[142]	IM_CH[141]	IM_CH[140]	IM_CH[139]	IM_CH[138]	IM_CH[137]	IM_CH[136]	IM_CH[135]	IM_CH[134]	IM_CH[133]	IM_CH[132]	IM_CH[131]	IM_CH[130]	IM_CH[129]	IM_CH[128]
	W																
SARADCB_ECIMR1	R	IM_CH[191]	IM_CH[190]	IM_CH[189]	IM_CH[188]	IM_CH[187]	IM_CH[186]	IM_CH[185]	IM_CH[184]	IM_CH[183]	IM_CH[182]	IM_CH[181]	IM_CH[180]	IM_CH[179]	IM_CH[178]	IM_CH[177]	IM_CH[176]
	W																
	R	IM_CH[175]	IM_CH[174]	IM_CH[173]	IM_CH[172]	IM_CH[171]	IM_CH[170]	IM_CH[169]	IM_CH[168]	IM_CH[167]	IM_CH[166]	IM_CH[165]	IM_CH[164]	IM_CH[163]	IM_CH[162]	IM_CH[161]	IM_CH[160]
	W																
SARADCB_ECDSR0	R	DS_CH[159]	DS_CH[158]	DS_CH[157]	DS_CH[156]	DS_CH[155]	DS_CH[154]	DS_CH[153]	DS_CH[152]	DS_CH[151]	DS_CH[150]	DS_CH[149]	DS_CH[148]	DS_CH[147]	DS_CH[146]	DS_CH[145]	DS_CH[144]
	W																

Table continues on the next page...

Table 38-22. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R																
	W	DS_CH[143]	DS_CH[142]	DS_CH[141]	DS_CH[140]	DS_CH[139]	DS_CH[138]	DS_CH[137]	DS_CH[136]	DS_CH[135]	DS_CH[134]	DS_CH[133]	DS_CH[132]	DS_CH[131]	DS_CH[130]	DS_CH[129]	DS_CH[128]
SARADCB_ECDSR1	R																
	W	DS_CH[191]	DS_CH[190]	DS_CH[189]	DS_CH[188]	DS_CH[187]	DS_CH[186]	DS_CH[185]	DS_CH[184]	DS_CH[183]	DS_CH[182]	DS_CH[181]	DS_CH[180]	DS_CH[179]	DS_CH[178]	DS_CH[177]	DS_CH[176]
	R																
	W	DS_CH[175]	DS_CH[174]	DS_CH[173]	DS_CH[172]	DS_CH[171]	DS_CH[170]	DS_CH[169]	DS_CH[168]	DS_CH[167]	DS_CH[166]	DS_CH[165]	DS_CH[164]	DS_CH[163]	DS_CH[162]	DS_CH[161]	DS_CH[160]
SARADCB_ECNCMR0	R																
	W	NCE_CH[159]	NCE_CH[158]	NCE_CH[157]	NCE_CH[156]	NCE_CH[155]	NCE_CH[154]	NCE_CH[153]	NCE_CH[152]	NCE_CH[151]	NCE_CH[150]	NCE_CH[149]	NCE_CH[148]	NCE_CH[147]	NCE_CH[146]	NCE_CH[145]	NCE_CH[144]
	R																
	W	NCE_CH[143]	NCE_CH[142]	NCE_CH[141]	NCE_CH[140]	NCE_CH[139]	NCE_CH[138]	NCE_CH[137]	NCE_CH[136]	NCE_CH[135]	NCE_CH[134]	NCE_CH[133]	NCE_CH[132]	NCE_CH[131]	NCE_CH[130]	NCE_CH[129]	NCE_CH[128]
SARADCB_ECNCMR1	R																
	W	NCE_CH[191]	NCE_CH[190]	NCE_CH[189]	NCE_CH[188]	NCE_CH[187]	NCE_CH[186]	NCE_CH[185]	NCE_CH[184]	NCE_CH[183]	NCE_CH[182]	NCE_CH[181]	NCE_CH[180]	NCE_CH[179]	NCE_CH[178]	NCE_CH[177]	NCE_CH[176]
	R																
	W	NCE_CH[175]	NCE_CH[174]	NCE_CH[173]	NCE_CH[172]	NCE_CH[171]	NCE_CH[170]	NCE_CH[169]	NCE_CH[168]	NCE_CH[167]	NCE_CH[166]	NCE_CH[165]	NCE_CH[164]	NCE_CH[163]	NCE_CH[162]	NCE_CH[161]	NCE_CH[160]
SARADCB_ECJCMR0	R																
	W	JCE_CH[159]	JCE_CH[158]	JCE_CH[157]	JCE_CH[156]	JCE_CH[155]	JCE_CH[154]	JCE_CH[153]	JCE_CH[152]	JCE_CH[151]	JCE_CH[150]	JCE_CH[149]	JCE_CH[148]	JCE_CH[147]	JCE_CH[146]	JCE_CH[145]	JCE_CH[144]
	R																
	W	JCE_CH[143]	JCE_CH[142]	JCE_CH[141]	JCE_CH[140]	JCE_CH[139]	JCE_CH[138]	JCE_CH[137]	JCE_CH[136]	JCE_CH[135]	JCE_CH[134]	JCE_CH[133]	JCE_CH[132]	JCE_CH[131]	JCE_CH[130]	JCE_CH[129]	JCE_CH[128]

Table continues on the next page...

Table 38-22. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADCB_ECJCMR1	R	JCE_CH[191]	JCE_CH[190]	JCE_CH[189]	JCE_CH[188]	JCE_CH[187]	JCE_CH[186]	JCE_CH[185]	JCE_CH[184]	JCE_CH[183]	JCE_CH[182]	JCE_CH[181]	JCE_CH[180]	JCE_CH[179]	JCE_CH[178]	JCE_CH[177]	JCE_CH[176]
	W																
	R	JCE_CH[175]	JCE_CH[174]	JCE_CH[173]	JCE_CH[172]	JCE_CH[171]	JCE_CH[170]	JCE_CH[169]	JCE_CH[168]	JCE_CH[167]	JCE_CH[166]	JCE_CH[165]	JCE_CH[164]	JCE_CH[163]	JCE_CH[162]	JCE_CH[161]	JCE_CH[160]
	W																
SARADCB_ECWSELR0	R	0	WSEL_CH135			0	WSEL_CH134			0	WSEL_CH133			0	WSEL_CH132		
	W																
	R	0	WSEL_CH131			0	WSEL_CH130			0	WSEL_CH129			0	WSEL_CH128		
	W																
SARADCB_ECWSELR1	R	0	WSEL_CH143			0	WSEL_CH142			0	WSEL_CH141			0	WSEL_CH140		
	W																
	R	0	WSEL_CH139			0	WSEL_CH138			0	WSEL_CH137			0	WSEL_CH136		
	W																
SARADCB_ECWSELR2	R	0	WSEL_CH151			0	WSEL_CH150			0	WSEL_CH149			0	WSEL_CH148		
	W																
	R	0	WSEL_CH147			0	WSEL_CH146			0	WSEL_CH145			0	WSEL_CH144		
	W																
SARADCB_ECWSELR3	R	0	WSEL_CH159			0	WSEL_CH158			0	WSEL_CH157			0	WSEL_CH156		
	W																
	R	0	WSEL_CH155			0	WSEL_CH154			0	WSEL_CH153			0	WSEL_CH152		
	W																
SARADCB_ECWSELR4	R	0	WSEL_CH167			0	WSEL_CH166			0	WSEL_CH165			0	WSEL_CH164		
	W																
	R	0	WSEL_CH163			0	WSEL_CH162			0	WSEL_CH161			0	WSEL_CH160		
	W																
SARADCB_ECWSELR5	R	0	WSEL_CH175			0	WSEL_CH174			0	WSEL_CH173			0	WSEL_CH172		
	W																
	R	0	WSEL_CH171			0	WSEL_CH170			0	WSEL_CH169			0	WSEL_CH168		
	W																
SARADCB_ECWSELR6	R	0	WSEL_CH183			0	WSEL_CH182			0	WSEL_CH181			0	WSEL_CH180		
	W																
	R	0	WSEL_CH179			0	WSEL_CH178			0	WSEL_CH177			0	WSEL_CH176		
	W																
SARADCB_ECWSELR7	R	0	WSEL_CH191			0	WSEL_CH190			0	WSEL_CH189			0	WSEL_CH188		
	W																

Table continues on the next page...

Table 38-22. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	WSEL_CH187			0	WSEL_CH186			0	WSEL_CH185			0	WSEL_CH184		
	W																
SARADCB_ECWENR0	R	WEN_CH[159]	WEN_CH[158]	WEN_CH[157]	WEN_CH[156]	WEN_CH[155]	WEN_CH[154]	WEN_CH[153]	WEN_CH[152]	WEN_CH[151]	WEN_CH[150]	WEN_CH[149]	WEN_CH[148]	WEN_CH[147]	WEN_CH[146]	WEN_CH[145]	WEN_CH[144]
	W																
	R	WEN_CH[143]	WEN_CH[142]	WEN_CH[141]	WEN_CH[140]	WEN_CH[139]	WEN_CH[138]	WEN_CH[137]	WEN_CH[136]	WEN_CH[135]	WEN_CH[134]	WEN_CH[133]	WEN_CH[132]	WEN_CH[131]	WEN_CH[130]	WEN_CH[129]	WEN_CH[128]
	W																
SARADCB_ECWENR1	R	WEN_CH[191]	WEN_CH[190]	WEN_CH[189]	WEN_CH[188]	WEN_CH[187]	WEN_CH[186]	WEN_CH[185]	WEN_CH[184]	WEN_CH[183]	WEN_CH[182]	WEN_CH[181]	WEN_CH[180]	WEN_CH[179]	WEN_CH[178]	WEN_CH[177]	WEN_CH[176]
	W																
	R	WEN_CH[175]	WEN_CH[174]	WEN_CH[173]	WEN_CH[172]	WEN_CH[171]	WEN_CH[170]	WEN_CH[169]	WEN_CH[168]	WEN_CH[167]	WEN_CH[166]	WEN_CH[165]	WEN_CH[164]	WEN_CH[163]	WEN_CH[162]	WEN_CH[161]	WEN_CH[160]
	W																
SARADCB_ECAWORR0	R	AWOR_CH[159]	AWOR_CH[158]	AWOR_CH[157]	AWOR_CH[156]	AWOR_CH[155]	AWOR_CH[154]	AWOR_CH[153]	AWOR_CH[152]	AWOR_CH[151]	AWOR_CH[150]	AWOR_CH[149]	AWOR_CH[148]	AWOR_CH[147]	AWOR_CH[146]	AWOR_CH[145]	AWOR_CH[144]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH[143]	AWOR_CH[142]	AWOR_CH[141]	AWOR_CH[140]	AWOR_CH[139]	AWOR_CH[138]	AWOR_CH[137]	AWOR_CH[136]	AWOR_CH[135]	AWOR_CH[134]	AWOR_CH[133]	AWOR_CH[132]	AWOR_CH[131]	AWOR_CH[130]	AWOR_CH[129]	AWOR_CH[128]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECAWORR1	R	AWOR_CH[191]	AWOR_CH[190]	AWOR_CH[189]	AWOR_CH[188]	AWOR_CH[187]	AWOR_CH[186]	AWOR_CH[185]	AWOR_CH[184]	AWOR_CH[183]	AWOR_CH[182]	AWOR_CH[181]	AWOR_CH[180]	AWOR_CH[179]	AWOR_CH[178]	AWOR_CH[177]	AWOR_CH[176]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-22. SARADC_B external channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	AWOR_CH[175]	AWOR_CH[174]	AWOR_CH[173]	AWOR_CH[172]	AWOR_CH[171]	AWOR_CH[170]	AWOR_CH[169]	AWOR_CH[168]	AWOR_CH[167]	AWOR_CH[166]	AWOR_CH[165]	AWOR_CH[164]	AWOR_CH[163]	AWOR_CH[162]	AWOR_CH[161]	AWOR_CH[160]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECMICR0	R	0	ICSEL_ECH152_159							0	ICSEL_ECH144_151						
	W																
	R	0	ICSEL_ECH136_143							0	ICSEL_ECH128_135						
	W																
SARADCB_ECMICR1	R	0	ICSEL_ECH184_191							0	ICSEL_ECH176_183						
	W																
	R	0	ICSEL_ECH168_175							0	ICSEL_ECH160_167						
	W																
SARADCB_ ECDR128-191ECDR128-211	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

38.2.2.9.2 SARADC_0 register definitions

Table 38-23. SARADC_0 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC0_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC0_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																

Table continues on the next page...

Table 38-23. SARADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC0_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC0_ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	EOC_CH[7]	EOC_CH[6]	EOC_CH[5]	EOC_CH[4]	0	0	0	0
	W									w1c	w1c	w1c	w1c				
SARADC0_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC0_ICIMR0	R																
	W																
	R	0	0	0	0	0	0	0	0	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	0	0	0	0
	W																
SARADC0_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC0_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC0_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-23. SARADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC0_ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DS_CH[7]	DS_CH[6]	DS_CH[5]	DS_CH[4]	0	0	0	0
	W																
SARADC0_WTHRHLR[0:3]	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC0_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC0_ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	NCE_CH[7]	NCE_CH[6]	NCE_CH[5]	NCE_CH[4]	0	0	0	0
	W																
SARADC0_ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	JCE_CH[7]	JCE_CH[6]	JCE_CH[5]	JCE_CH[4]	0	0	0	0
	W																
SARADC0_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC0_ICDR4-7	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC0_ICWSELR0	R	0	0	WSEL_CH7		0	0	WSEL_CH6		0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																

Table continues on the next page...

Table 38-23. SARADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC0_ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WEN_CH[7]	WEN_CH[6]	WEN_CH[5]	WEN_CH[4]	0	0	0	0
	W																
SARADC0_ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	AWOR_CH[7]	AWOR_CH[6]	AWOR_CH[5]	AWOR_CH[4]	0	0	0	0
	W									w1c	w1c	w1c	w1c				

38.2.2.9.3 SARADC_1 register definitions

Table 38-24. SARADC_1 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC1_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC1_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC1_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-24. SARADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC1_ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOC_CH[15]	EOC_CH[14]	EOC_CH[13]	EOC_CH[12]	EOC_CH[11]	EOC_CH[10]	EOC_CH[9]	EOC_CH[8]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								
SARADC1_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC1_ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	0	0	0	0	0	0	0	0
	W																
SARADC1_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC1_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC1_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC1_ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 38-24. SARADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	DS_CH[15]	DS_CH[14]	DS_CH[13]	DS_CH[12]	DS_CH[11]	DS_CH[10]	DS_CH[9]	DS_CH[8]	0	0	0	0	0	0	0	0
	W																
SARADC1_WTHRHLR[0:3]	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC1_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC1_ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NCE_CH[15]	NCE_CH[14]	NCE_CH[13]	NCE_CH[12]	NCE_CH[11]	NCE_CH[10]	NCE_CH[9]	NCE_CH[8]	0	0	0	0	0	0	0	0
	W																
SARADC1_ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	JCE_CH[15]	JCE_CH[14]	JCE_CH[13]	JCE_CH[12]	JCE_CH[11]	JCE_CH[10]	JCE_CH[9]	JCE_CH[8]	0	0	0	0	0	0	0	0
	W																
SARADC1_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC1_ICDR8-15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC1_ICWSELR1	R	0	0	WSEL_CH15		0	0	WSEL_CH14		0	0	WSEL_CH13		0	0	WSEL_CH12	
	W																
	R	0	0	WSEL_CH11		0	0	WSEL_CH10		0	0	WSEL_CH9		0	0	WSEL_CH8	
	W																
SARADC1_ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 38-24. SARADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	WEN_CH[15]	WEN_CH[14]	WEN_CH[13]	WEN_CH[12]	WEN_CH[11]	WEN_CH[10]	WEN_CH[9]	WEN_CH[8]	0	0	0	0	0	0	0	0
	W																
SARADC1_ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH[15]	AWOR_CH[14]	AWOR_CH[13]	AWOR_CH[12]	AWOR_CH[11]	AWOR_CH[10]	AWOR_CH[9]	AWOR_CH[8]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								

38.2.2.9.4 SARADC_2 register definitions

Table 38-25. SARADC_2 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC2_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC2_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC2_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-25. SARADC_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC2_ICIPR0	R	0	0	0	0	0	0	0	0	EOC_CH[23]	EOC_CH[22]	0	0	EOC_CH[19]	EOC_CH[18]	EOC_CH[17]	EOC_CH[16]
	W									w1c	w1c			w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC2_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC2_ICIMR0	R	0	0	0	0	0	0	0	0	IM_CH[23]	IM_CH[22]	0	0	IM_CH[19]	IM_CH[18]	IM_CH[17]	IM_CH[16]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC2_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC2_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC2_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC2_ICDSR0	R	0	0	0	0	0	0	0	0	DS_CH[23]	DS_CH[22]	0	0	DS_CH[19]	DS_CH[18]	DS_CH[17]	DS_CH[16]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 38-25. SARADC_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
SARADC2_WTHRHLR[0:3]	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC2_CTR[0:3]	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC2_ICNCMR0	R	0	0	0	0	0	0	0	0	NCE_CH[23]	NCE_CH[22]	0	0	NCE_CH[19]	NCE_CH[18]	NCE_CH[17]	NCE_CH[16]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC2_ICJCMR0	R	0	0	0	0	0	0	0	0	JCE_CH[23]	JCE_CH[22]	0	0	JCE_CH[19]	JCE_CH[18]	JCE_CH[17]	JCE_CH[16]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC2_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC2_ICDR16-19, ICDR[22:23]	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC2_ICWSELR2	R	0	0	WSEL_CH23		0	0	WSEL_CH22		0	0	0	0	0	0	0	0
	W																
	R	0	0	WSEL_CH19		0	0	WSEL_CH18		0	0	WSEL_CH17		0	0	WSEL_CH16	
	W																
SARADC2_ICWENR0	R	0	0	0	0	0	0	0	0	WEN_CH[23]	WEN_CH[22]	0	0	WEN_CH[19]	WEN_CH[18]	WEN_CH[17]	WEN_CH[16]
	W																

Table continues on the next page...

Table 38-25. SARADC_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC2_ICAWORR0	R	0	0	0	0	0	0	0	0	AWOR_CH[23]	AWOR_CH[22]	0	0	AWOR_CH[19]	AWOR_CH[18]	AWOR_CH[17]	AWOR_CH[16]
	W									w1c	w1c			w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

38.2.2.9.5 SARADC_3 register definitions

Table 38-26. SARADC_3 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC3_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC3_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC3_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-26. SARADC_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC3_ICIPR0	R	EOC_CH[31]	EOC_CH[30]	EOC_CH[29]	EOC_CH[28]	EOC_CH[27]	EOC_CH[26]	0	0	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c	w1c	w1c										
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC3_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC3_ICIMR0	R							0	0	0	0	0	0	0	0	0	0
	W	IM_CH[31]	IM_CH[30]	IM_CH[29]	IM_CH[28]	IM_CH[27]	IM_CH[26]										
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC3_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC3_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC3_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC3_ICDSR0	R							0	0	0	0	0	0	0	0	0	0
	W	DS_CH[31]	DS_CH[30]	DS_CH[29]	DS_CH[28]	DS_CH[27]	DS_CH[26]										
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 38-26. SARADC_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
SARADC3_WTHRHLR[0:3]	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC3_CTR[0:3]	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC3_ICNCMR0	R							0	0	0	0	0	0	0	0	0	0
	W	NCE_CH[31]	NCE_CH[30]	NCE_CH[29]	NCE_CH[28]	NCE_CH[27]	NCE_CH[26]										
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC3_ICJCMR0	R							0	0	0	0	0	0	0	0	0	0
	W	JCE_CH[31]	JCE_CH[30]	JCE_CH[29]	JCE_CH[28]	JCE_CH[27]	JCE_CH[26]										
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC3_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC3_ICDR[26:31]	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC3_ICWSELR3	R	0	0	WSEL_CH31		0	0	WSEL_CH30		0	0	WSEL_CH29		0	0	WSEL_CH28	
	W																
	R	0	0	WSEL_CH27		0	0	WSEL_CH26		0	0	0	0	0	0	0	0
	W																
SARADC3_ICWENR0	R							0	0	0	0	0	0	0	0	0	0
	W	WEN_CH[31]	WEN_CH[30]	WEN_CH[29]	WEN_CH[28]	WEN_CH[27]	WEN_CH[26]										

Table continues on the next page...

Table 38-26. SARADC_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC3_ICAWORR0	R	AWOR_CH[31]	AWOR_CH[30]	AWOR_CH[29]	AWOR_CH[28]	AWOR_CH[27]	AWOR_CH[26]	0	0	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c	w1c	w1c										
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

38.2.2.9.6 SARADC_4 register definitions

Table 38-27. SARADC_4 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC4_MCR	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W	OWREN	WLSIDE	MODE		NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ			
SARADC4_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC4_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC4_ICIPR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-27. SARADC_4 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	EOC_CH[35]	EOC_CH[34]	EOC_CH[33]	EOC_CH[32]
	W													w1c	w1c	w1c	w1c
SARADC4_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC4_ICIMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	IM_CH[35]	IM_CH[34]	IM_CH[33]	IM_CH[32]
	W																
SARADC4_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC4_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC4_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC4_ICDSR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	DS_CH[35]	DS_CH[34]	DS_CH[33]	DS_CH[32]
	W																
SARADC4_WTHRHLR[0:3]	R	0	0	0	0	THRH											

Table continues on the next page...

Table 38-27. SARADC_4 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	0	0	0	0	THRL											
	W																
SARADC4_CTR[0:3]	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC4_ICNCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	NCE_CH[35]	NCE_CH[34]	NCE_CH[33]	NCE_CH[32]
	W																
SARADC4_ICJCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JCE_CH[35]	JCE_CH[34]	JCE_CH[33]	JCE_CH[32]
	W																
SARADC4_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC4_ICDR[32:35]	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC4_ICWSELR4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	WSEL_CH35		0	0	WSEL_CH34		0	0	WSEL_CH33		0	0	WSEL_CH32	
	W																
SARADC4_ICWENR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	WEN_CH[35]	WEN_CH[34]	WEN_CH[33]	WEN_CH[32]
	W																

Table continues on the next page...

Table 38-27. SARADC_4 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC4_ICAWORR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH[35]	AWOR_CH[34]	AWOR_CH[33]	AWOR_CH[32]
	W													w1c	w1c	w1c	w1c

38.2.2.9.7 SARADC_5 register definitions

Table 38-28. SARADC_5 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC5_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC5_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC5_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC5_ICIPR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-28. SARADC_5 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	EOC_CH[43]	EOC_CH[42]	EOC_CH[41]	EOC_CH[40]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
SARADC5_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC5_ICIMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	IM_CH[43]	IM_CH[42]	IM_CH[41]	IM_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC5_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC5_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC5_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC5_ICDSR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	DS_CH[43]	DS_CH[42]	DS_CH[41]	DS_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC5_	R	0	0	0	0	THRH											

Table continues on the next page...

Table 38-28. SARADC_5 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
WTHRHLR0–WTHRHLR3	W					THRL											
	R	0	0	0	0												
	W																
SARADC5_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC5_ICNCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	NCE_CH[43]	NCE_CH[42]	NCE_CH[41]	NCE_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC5_ICJCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	JCE_CH[43]	JCE_CH[42]	JCE_CH[41]	JCE_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC5_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC5_ICDR40-43, ICDR48-51	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC5_ICWSELR5	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH43			0	WSEL_CH42			0	WSEL_CH41			0	WSEL_CH40		
	W																
SARADC5_ICWSELR6	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC5_ICWENR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-28. SARADC_5 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	WEN_CH[43]	WEN_CH[42]	WEN_CH[41]	WEN_CH[40]	0	0	0	0	0	0	0	0
	W																
SARADC5_ICAWORR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	AWOR_CH[43]	AWOR_CH[42]	AWOR_CH[41]	AWOR_CH[40]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								

38.2.2.9.8 SARADC_6 register definitions

Table 38-29. SARADC_6 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC6_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL			ABORT	ABORTF		0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC6_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC6_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-29. SARADC_6 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC6_ICIPR1	R	0	0	0	0	0	0	0	0	0	0	0	0	EOC_CH[51]	EOC_CH[50]	EOC_CH[49]	EOC_CH[48]
	W													w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC6_ICIMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	IM_CH[51]	IM_CH[50]	IM_CH[49]	IM_CH[48]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC6_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC6_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC6_ICDSR1	R	0	0	0	0	0	0	0	0	0	0	0	0	DS_CH[51]	DS_CH[50]	DS_CH[49]	DS_CH[48]
	W																

Table continues on the next page...

Table 38-29. SARADC_6 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_WTHRHLR0–WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC6_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC6_ICNCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	NCE_CH[51]	NCE_CH[50]	NCE_CH[49]	NCE_CH[48]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_ICJCMR1	R	0	0	0	0	0	0	0	0	0	0	0	0	JCE_CH[51]	JCE_CH[50]	JCE_CH[49]	JCE_CH[48]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC6_ICDR40-43,ICDR48-51	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC6_ICWSELR5	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_ICWSELR6	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH51			0	WSEL_CH50			0	WSEL_CH49			0	WSEL_CH48		

Table continues on the next page...

Table 38-29. SARADC_6 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
SARADC6_ICWENR1	R	0	0	0	0	0	0	0	0	0	0	0	0	WEN_CH[51]	WEN_CH[50]	WEN_CH[49]	WEN_CH[48]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC6_ICAWORR1	R	0	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH[51]	AWOR_CH[50]	AWOR_CH[49]	AWOR_CH[48]
	W													w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

38.2.2.9.9 SARADC_7 register definitions

Table 38-30. SARADC_7 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC7_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC7_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-30. SARADC_7 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC7_ICIPR1	R	0	0	0	0	EOC_CH[59]	EOC_CH[58]	EOC_CH[57]	EOC_CH[56]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC7_ICIMR1	R	0	0	0	0	IM_CH[59]	IM_CH[58]	IM_CH[57]	IM_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC7_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC7_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																

Table continues on the next page...

Table 38-30. SARADC_7 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_ICDSR1	R	0	0	0	0	DS_CH[59]	DS_CH[58]	DS_CH[57]	DS_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_WTHRHLR0–WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC7_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC7_ICNCMR1	R	0	0	0	0	NCE_CH[59]	NCE_CH[58]	NCE_CH[57]	NCE_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_ICJCMR1	R	0	0	0	0	JCE_CH[59]	JCE_CH[58]	JCE_CH[57]	JCE_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC7_ICDR56-59	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC7_ICWSELR7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH59			0	WSEL_CH58			0	WSEL_CH57			0	WSEL_CH56		
	W																

Table continues on the next page...

Table 38-30. SARADC_7 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC7_ICWENR1	R	0	0	0	0	WEN_CH[59]	WEN_CH[58]	WEN_CH[57]	WEN_CH[56]	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC7_ICAWORR1	R	0	0	0	0	AWOR_CH[59]	AWOR_CH[58]	AWOR_CH[57]	AWOR_CH[56]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

38.2.2.9.10 SARADC_8 register definitions

Table 38-31. SARADC_8 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC8_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC8_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC8_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH

Table continues on the next page...

Table 38-31. SARADC_8 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W													w1c	w1c	w1c	w1c
SARADC8_ICIPR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	EOC_CH[67]	EOC_CH[66]	EOC_CH[65]	EOC_CH[64]
	W													w1c	w1c	w1c	w1c
SARADC8_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC8_ICIMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	IM_CH[67]	IM_CH[66]	IM_CH[65]	IM_CH[64]
	W																
SARADC8_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC8_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC8_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC8_ICDSR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-31. SARADC_8 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	DS_CH[67]	DS_CH[66]	DS_CH[65]	DS_CH[64]
	W																
SARADC8_WTHRHLR0–WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC8_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC8_ICNCMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	NCE_CH[67]	NCE_CH[66]	NCE_CH[65]	NCE_CH[64]
	W																
SARADC8_ICJCMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JCE_CH[67]	JCE_CH[66]	JCE_CH[65]	JCE_CH[64]
	W																
SARADC8_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC8_ICDR64-67	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC8_ICWSELR7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH67			0	WSEL_CH66			0	WSEL_CH65			0	WSEL_CH64		
	W																
SARADC8_ICWENR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-31. SARADC_8 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	WEN_CH[67]	WEN_CH[66]	WEN_CH[65]	WEN_CH[64]
	W																
SARADC8_ICAWORR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH[67]	AWOR_CH[66]	AWOR_CH[65]	AWOR_CH[64]
	W													w1c	w1c	w1c	w1c

38.2.2.9.11 SARADC_9 register definitions

Table 38-32. SARADC_9 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC9_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC9_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC9_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c
SARADC9_ICIPR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 38-32. SARADC_9 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	0	0	0	0	EOC_CH[75]	EOC_CH[74]	EOC_CH[73]	EOC_CH[72]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								
SARADC9_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC9_ICIMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	IM_CH[75]	IM_CH[74]	IM_CH[73]	IM_CH[72]	0	0	0	0	0	0	0	0
	W																
SARADC9_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC9_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC9_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC9_ICDSR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	DS_CH[75]	DS_CH[74]	DS_CH[73]	DS_CH[72]	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-32. SARADC_9 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC9_WTHRHLR0–WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC9_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC9_ICNCMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	NCE_CH[75]	NCE_CH[74]	NCE_CH[73]	NCE_CH[72]	0	0	0	0	0	0	0	0
	W																
SARADC9_ICJCMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	JCE_CH[75]	JCE_CH[74]	JCE_CH[73]	JCE_CH[72]	0	0	0	0	0	0	0	0
	W																
SARADC9_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC9_ICDR72-75	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC9_ICWSELR7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH75			0	WSEL_CH74			0	WSEL_CH73			0	WSEL_CH72		
	W																
SARADC9_ICWENR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 38-32. SARADC_9 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	WEN_CH[75]	WEN_CH[74]	WEN_CH[73]	WEN_CH[72]	0	0	0	0	0	0	0	0
	W																
SARADC9_ICAWORR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	AWOR_CH[75]	AWOR_CH[74]	AWOR_CH[73]	AWOR_CH[72]	0	0	0	0	0	0	0	0
	W					w1c	w1c	w1c	w1c								

38.2.2.9.12 SARADC_10 register definitions

Table 38-33. SARADC_10 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC10_MCR	R	OWREN	WLSIDE	MODE	0	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORTCHAIN	ABORT	0	FRZ	0	0	EDCSELF	PWDN
	W																
SARADC10_MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
SARADC10_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	JEOC	JECH	NEOC	NECH
	W													w1c	w1c	w1c	w1c

Table continues on the next page...

Table 38-33. SARADC_10 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SARADC10_ICIPR2	R	0	0	0	0	0	0	0	0	0	0	0	0	EOC_CH[83]	EOC_CH[82]	EOC_CH[81]	EOC_CH[80]
	W													w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC10_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
SARADC10_ICIMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	IM_CH[83]	IM_CH[82]	IM_CH[81]	IM_CH[80]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC10_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC10_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADC10_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
SARADC10_ICDSR2	R	0	0	0	0	0	0	0	0	0	0	0	0	DS_CH[83]	DS_CH[82]	DS_CH[81]	DS_CH[80]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 38-33. SARADC_10 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
SARADC10_ WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC10_CTR0-3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
SARADC10_ICNCMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	NCE_CH[83]	NCE_CH[82]	NCE_CH[81]	NCE_CH[80]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC10_ICJCMR2	R	0	0	0	0	0	0	0	0	0	0	0	0	JCE_CH[83]	JCE_CH[82]	JCE_CH[81]	JCE_CH[80]
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC10_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
SARADC10_ICDR80-83	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC10_ICWSELR7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH83			0	WSEL_CH82			0	WSEL_CH81			0	WSEL_CH80		
	W																
SARADC10_ICWENR2	R	0	0	0	0	0	0	0	0	0	0	0	0	WEN_CH[83]	WEN_CH[82]	WEN_CH[81]	WEN_CH[80]
	W																

Table continues on the next page...

Table 38-33. SARADC_10 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC10_ICAWORR2	R	0	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH[83]	AWOR_CH[82]	AWOR_CH[81]	AWOR_CH[80]
	W													w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Chapter 39
Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface

39.1 Introduction

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides accurate conversion data for a wide range of applications.

39.2 Overview

The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, etc. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. It also provides external modulator support and keeps the internal modulator in low consumption mode. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

The following figure depicts the block diagram of an SDADC and its digital control block.

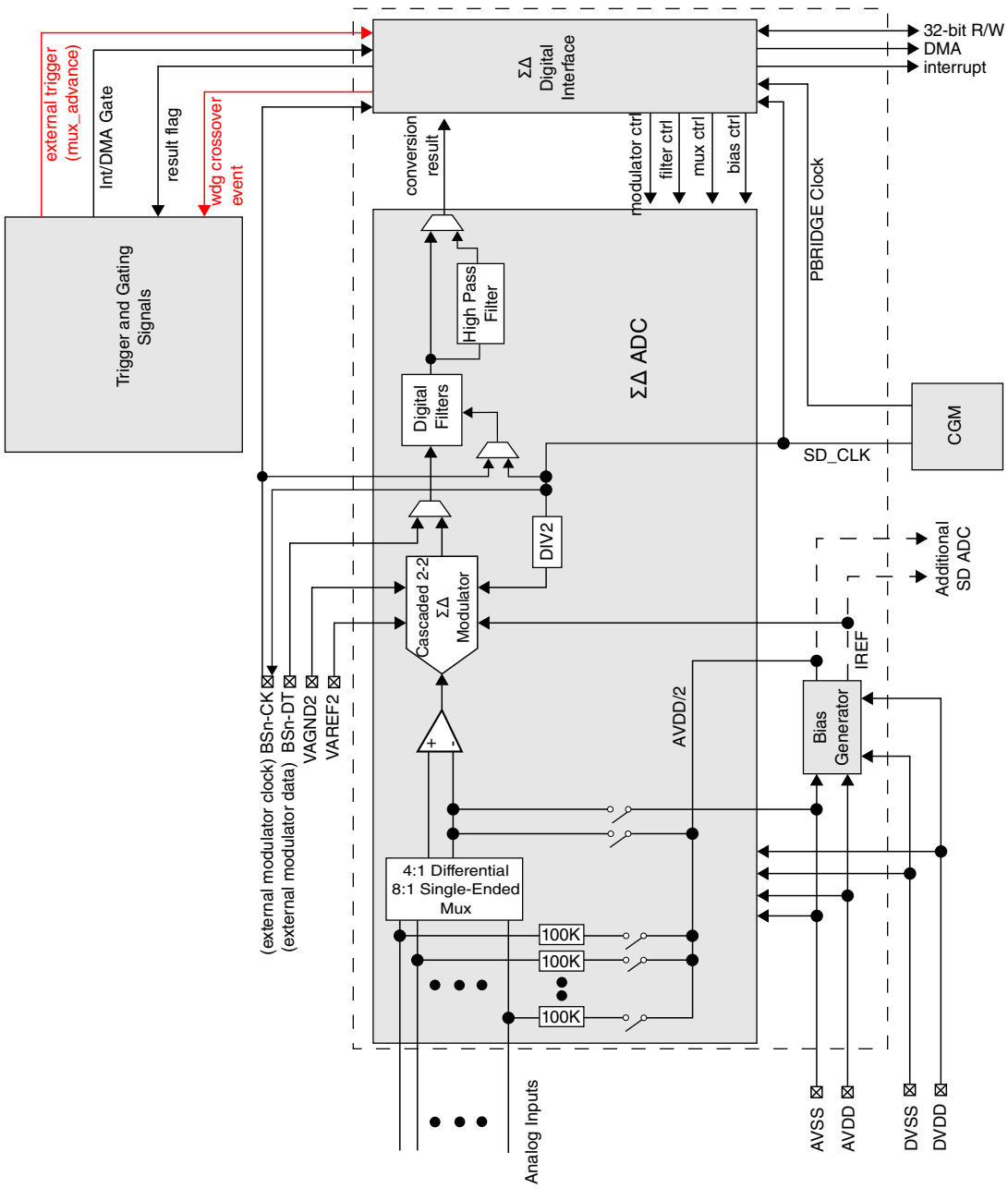


Figure 39-1. SDADC block diagram

39.3 Features

The features of the SDADC are as follows:

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation

    - Configurable trigger sources: hardware or software
    - Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Programmable gain for analog inputs
- Optional external modulator support
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing up to 16 datawords of 16-bit converted data
- Interrupt/DMA request generation based on various conditions:
    - Data buffer above threshold
    - Data buffer overrun
    - Watchdog (WDG) crossover event
- Programmable threshold range watchdog support
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request

39.4 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

39.4.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC modulator. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

39.4.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

39.4.3 External modulator mode

This mode is entered by asserting the MCR[EMSEL] bit. In this mode, the data stream and clock outputs from the external modulator available on EMDATA/EMCLK input pins are routed directly to the SDADC block. The internal modulator is bypassed in this mode. Only the digital filters of the SDADC will be used for result calculation. The source for the external modulator clock can be the internal on-chip clock source or the direct clock output coming from the external modulator. The external pin function control for these pins must be configured in the integration logic on a device, and is not included in the SDADC digital interface.

39.5 External signal description

The following table shows the list of signals driven by external pins. See the chip-specific details about SDADC signals and package pinouts for information about signal muxing and availability external to the chip.

Table 39-1. Signal properties

Name	Function	I/O	Reset	Pull-up
AN[0:7]	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
EMDATA	External modulator data input	I	0	—
EMCLK	External modulator clock input	I	0	—
VREFP	Voltage reference for SDADC	I	—	—
VREFN	Ground reference for SDADC	I	—	—
AVDD	Analog voltage supply for SDADC	I	—	—
AVSS	Analog ground supply for SDADC	I	—	—
DVDD	Digital voltage supply for SDADC	I	—	—
DVSS	Digital ground supply for SDADC	I	—	—

39.6 Memory map and register definition

This section provides the memory map for the SDADC block.

SDADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (SDADC_MCR)	32	R/W	0000_0000h	39.6.1/1561

Table continues on the next page...

SDADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Channel Selection Register (SDADC_CSR)	32	R/W	0000_0000h	39.6.2/1565
8	Reset Key Register (SDADC_RKR)	32	R/W	0000_A50Fh	39.6.3/1566
C	Status Flag Register (SDADC_SFR)	32	R/W	0000_0100h	39.6.4/1567
10	Request Select and Enable Register (SDADC_RSER)	32	R/W	0000_0000h	39.6.5/1569
14	Output Settling Delay Register (SDADC_OSDR)	32	R/W	0000_0000h	39.6.6/1571
18	FIFO Control Register (SDADC_FCR)	32	R/W	0000_0006h	39.6.7/1572
1C	Software Trigger Key Register (SDADC_STKR)	32	R/W	0000_0000h	39.6.8/1573
20	Converted Data Register (SDADC_CDR)	32	R	0000_0000h	39.6.9/1573
24	WDG Threshold Register (SDADC_WTHHLR)	32	R/W	0000_0000h	39.6.10/ 1574

39.6.1 Module Configuration Register (SDADC_MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			PDR					0	PGAN			Reserved	Reserved	EMSEL	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGEN	TRIGEDSEL		TRIGEN	TRIGSEL				FRZ	0		VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_MCR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PDR	Programmable Decimation Rate

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
	This field selects the over-sampling ratio to be applied to support different passbands with a fixed input sampling clock. The output data rate f_d is equal to f_s / (2xOSR), where fs is the input sampling clock frequency.When external modulator is selected, the output data rate is independent of this field and is fixed to f_s / 256. 00000 OSR = 24 00001 OSR = 28 00010 OSR = 32 00011 OSR = 36 00100 OSR = 40 00101 OSR = 44 00110 OSR = 48 00111 OSR = 56 01000 OSR = 64 01001 OSR = 72 01010 OSR = 75 01011 OSR = 80 01100 OSR = 88 01101 OSR = 96 01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 - 11111 Reserved
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. **NOTE:** When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
12 Reserved	This field is reserved. This field is reserved and always has the value 0. Only the reset value shall be written to this register field.

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
13 Reserved	This field is reserved. This field is reserved and always has the value 0. Only the reset value shall be written to this register field.
14 EMSEL	External Modulator Selection 0 External modulator data and clock inputs are ignored 1 External modulator data stream and clock inputs are provided to SDADC
15 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled 1 High-pass (DC removal) filter is enabled
16 WDGEN	Watchdog Enable This bit enables the WDG monitor. 0 WDG is disabled 1 WDG is enabled
17–18 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected 01 Rising edge of trigger input is selected 10 Both edges of trigger input are selected 11 Both edges of trigger input are selected
19 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled (MCR[EN] = 0) or the external modulator is selected (MCR[EMSEL] = 1). 0 Trigger input is disabled 1 Trigger input is enabled
20–23 TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware triggered conversions. 0000 Input trigger 0 is selected. See chip-specific information for trigger source. 0001 Input trigger 1 is selected. See chip-specific information for trigger source. 0010 Input trigger 2 is selected. See chip-specific information for trigger source. 0011 Input trigger 3 is selected. See chip-specific information for trigger source. 0100 Input trigger 4 is selected. See chip-specific information for trigger source. 0101 Input trigger 5 is selected. See chip-specific information for trigger source. 0110 Input trigger 6 is selected. See chip-specific information for trigger source. 0111 Input trigger 7 is selected. See chip-specific information for trigger source. 1000 Input trigger 8 is selected. See chip-specific information for trigger source. 1001 Input trigger 9 is selected. See chip-specific information for trigger source. 1010 MUX_ADV 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

Table continues on the next page...

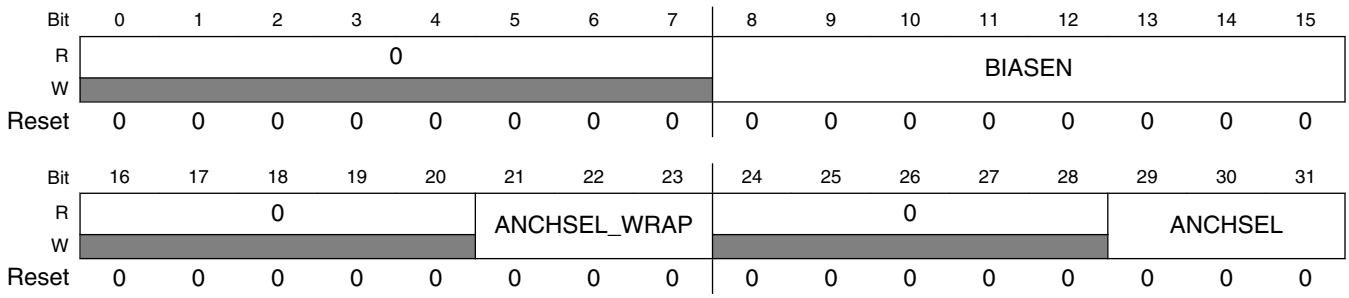
SDADC_MCR field descriptions (continued)

Field	Description
24 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the chip enters debug mode. When the chip enters debug mode, further conversions by the SDADC analog block will be stopped, and converted data output from the SDADC analog block are ignored. The ongoing channel conversion will be completed and the converted data is stored. When the chip exits debug mode, SDADC resumes the job that was left before halting the conversions. 0 Conversions are not stopped 1 Conversions are stopped
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 VCOMSEL	Common Voltage Bias Selection This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode (MODE = 1). 0 Negative input terminal is biased with VREFN 1 Negative input terminal is biased with VREFP/2 (half-scale bias)
28 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels. 0 Wraparound mechanism disabled (in this case the default software control mechanism will be enabled) 1 Wraparound mechanism enabled
29 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled
30 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
31 EN	Enable for SDADC block 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

39.6.2 Channel Selection Register (SDADC_CSR)

The Channel Selection Register (CSR) selects analog inputs AN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 kOhm resistance.

Address: 0h base + 4h offset = 4h



SDADC_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 BIASEN	Bias enable If BIASEN[i] = 1, analog input AN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 kOhm resistor.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 ANCHSEL_ WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it will wrap around back to initial value 000. This value should be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 ANCHSEL	Analog Channel Selection(ANCHSEL) Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to the following table. Wraparound Mode: • This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter will start incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)). • If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter will be updated with the value of ANCHSEL - the next trigger will act on the updated channel, and the next increment will be on the updated channel. Here, the value of ANCHSEL chosen should be < ANCHSEL_WRAP; otherwise it will be ignored by hardware. • Wraparound will always be from initial counter value 000.

Table continues on the next page...

SDADC_CSR field descriptions (continued)

Field	Description				
	MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)
	1	0	000	AN[0]	VREFN
			001	AN[1]	VREFN
			010	AN[2]	VREFN
			011	AN[3]	VREFN
			100	AN[4]	VREFN
			101	AN[5]	VREFN
			110	AN[6]	VREFN
			111	AN[7]	VREFN
		1	000	AN[0]	VREFP/2
			001	AN[1]	VREFP/2
			010	AN[2]	VREFP/2
			011	AN[3]	VREFP/2
			100	AN[4]	VREFP/2
			101	AN[5]	VREFP/2
			110	AN[6]	VREFP/2
			111	AN[7]	VREFP/2
	0	0/1	000	AN[0]	AN[1]
			001	AN[2]	AN[3]
			010	AN[4]	AN[5]
			011	AN[6]	AN[7]
			100	VREFN	VREFN
			101	VREFP/2	VREFP/2
			110	VREFP	VREFN
			111	VREFN	VREFP

39.6.3 Reset Key Register (SDADC_RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RESET_KEY															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

SDADC_RKR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RESET_KEY	Reset Key This field, when written with 5AF0h, is used to generate the reset for the SDADC block and to reload the internal counterwith the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access will always return the inverted key.

39.6.4 Status Flag Register (SDADC_SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing '1' to those bits. Writing a '0' to a flag has no effect.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													ANCHSEL_CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							DFEF	0			WTHH	WTHL	CDVF	DFORF	DFFF
W												w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SDADC_SFR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 ANCHSEL_CNT	Analog Channel Selection Counter(ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 DFEF	Data FIFO Empty Flag This field is asserted when there is no dataword in the FIFO. 0 Data FIFO is not empty 1 Data FIFO is empty
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 WTHH	Watchdog Upper Threshold Cross Over Event This field is set when Converted Data crosses Upper Threshold of WDG. It can be cleared by software writing '1' to clear. **NOTE:** This bit is also cleared by an acknowledgement from the DMA module when all of the following conditions are met: • SDADC_RSER[WTHDIRE]=1 (When this field is set by the occurrence of a watchdog upper threshold cross over event, a request is generated) • SDADC_SFR[WTHDIRS]=1 (The request generated when this field is set is a DMA request) • SDADC_RSER[GDIGE]=0 OR, if SDADC_RSER[GDIGE]=1, the external gating signal is asserted 0 Watchdog Upper Threshold Cross Over Event did not occur 1 Watchdog Upper Threshold Cross Over Event occurred
28 WTHL	Watchdog Lower Threshold Cross Over Event This field is set when Converted Data crosses Lower Threshold of WDG. It can be cleared by software writing '1' to clear. **NOTE:** This bit is also cleared by an acknowledgement from the DMA module when all of the following conditions are met: • SDADC_RSER[WTHDIRE]=1 (When this field is set by the occurrence of a watchdog lower threshold cross over event, a request is generated) • SDADC_SFR[WTHDIRS]=1 (The request generated when this field is set is a DMA request) • SDADC_RSER[GDIGE]=0 OR, if SDADC_RSER[GDIGE]=1, the external gating signal is asserted 0 Watchdog Lower Threshold Cross Over Event did not occur 1 Watchdog Lower Threshold Cross Over Event occurred
29 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or is corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. Any write access to the registers MCR and CSR will reset this flag. This flag is set automatically after an internal timer counts a number of output clock fd clock cycles specified by the value of SDADC_OSDR[OSD].

Table continues on the next page...

SDADC_SFR field descriptions (continued)

Field	Description
	0 Data output from SDADC is not valid 1 Data output from SDADC is valid
30 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted datawords. Software can clear this bit by writing '1' to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further datawords will not be received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred
31 DFFF	Data FIFO Full Flag This bit is set when the number of converted datawords in the FIFO is equal to or more than the number indicated by FCR[FTHLD]. Software can clear this bit by writing '1' to it. This bit is also cleared by an acknowledgement from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, datawords will continue to be received until an overrun condition occurs. Datawords will not be received if the converted data is not valid (CDVF = 0). **NOTE:** Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to also clear the RSER[DFFDIRE] bit alongwith. Similarly if DMA or software is too slow in not reading the FIFO datawords resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both these action will ensure safe operation. 0 The number of datawords in FIFO is less than the number indicated by FCR[FTHLD] 1 The number of datawords in FIFO is equal to or greater than the number indicated by FCR[FTHLD] at some point in time since this flag was last cleared

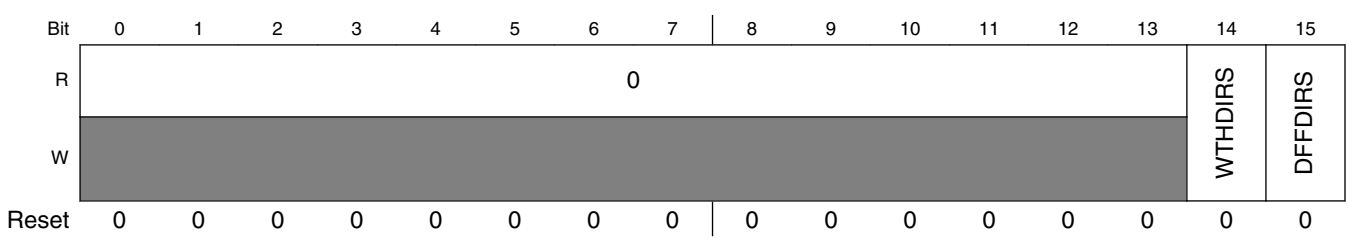
39.6.5 Request Select and Enable Register (SDADC_RSER)

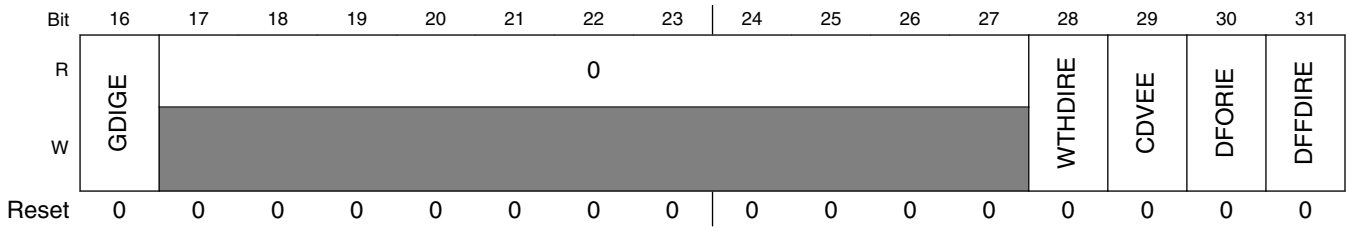
The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Refer to the field descriptions for the type of requests that are supported.

Address: 0h base + 10h offset = 10h





SDADC_RSER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 WTHDIRS	WDG Threshold Cross Over Event DMA/Interrupt Request Select When SFR[WTHH/WTHL] and RSER[WTHDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
15 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and SFR[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
16 GDIGE	Global DMA/Interrupt Gating Enable This field determines whether all module interrupts and DMA requests need to be gated by an external gating signal or not. 0 No impact of external gating signal on module DMA/interrupt requests 1 All module DMA/interrupt requests are qualified only when external gating signal is asserted
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 WTHDIRE	WDG Threshold Cross Over Event DMA/Interrupt Request Enable This field enables the SFR[WTHH/WTHL] field to generate a request. The WTHDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled on WDG Threshold Cross Over Event 1 Interrupt/DMA request is enabled on WDG Threshold Cross Over Event
29 CDVEE	Conversion Data Valid Event Enable The status of SFR[CDVF] gated with RSER[CDVEE] is passed on as an event on “conversion_flag" output port, which can be used at SoC level for indication of valid data conversion window. Once enabled, the event output will remain asserted as long as valid data conversions are ongoing. 0 Event output disabled 1 Event output assertion/deassertion based on the SFR[CDVF] field
30 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GDIGE field.

Table continues on the next page...

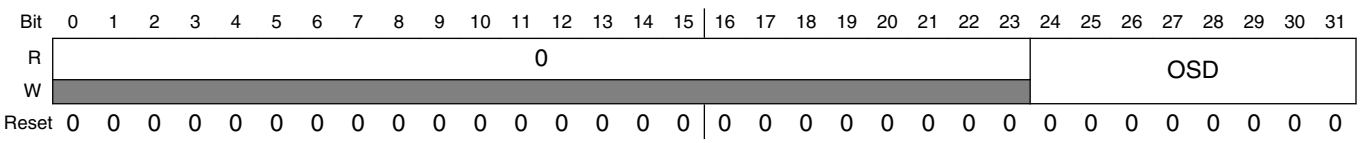
SDADC_RSER field descriptions (continued)

Field	Description
	0 Interrupt request is disabled when data FIFO overrun condition occurs 1 Interrupt request is enabled when data FIFO overrun condition occurs
31 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled when data FIFO full condition occurs 1 Interrupt/DMA request is enabled when data FIFO full condition occurs

39.6.6 Output Settling Delay Register (SDADC_OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

Address: 0h base + 14h offset = 14h



SDADC_OSDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OSD	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock fd is equal to f_s / (2xOSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output clock is fixed to f_s / 256. See the electrical specifications for the settling delays. **NOTE:** Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.

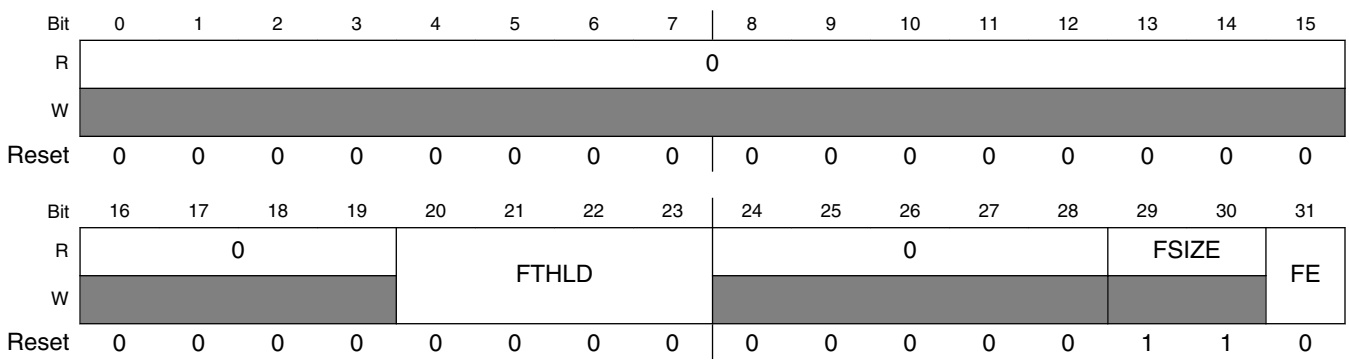
39.6.7 FIFO Control Register (SDADC_FCR)

The FIFO Control Register (FCR) provides the ability to enable or disable the FIFO functionality, and to control the amount of available FIFO depth to be used.

NOTE

1. It is important to note that the FIFO depth is set at the factory and cannot be changed but the amount of available FIFO space actually used can be set by specifying the value in the FTHLD field. The FSIZE field indicates the amount of FIFO space available. By setting the value of FTHLD to be less than the FIFO depth indicated by FSIZE, the FIFO is considered to be full when the number of data words in the FIFO is equal or greater than FTHLD.
2. This register may be read at any time, but should only be written when SDADC is disabled (MCR[EN] = 0).

Address: 0h base + 18h offset = 18h



SDADC_FCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 FTHLD	FIFO Threshold When the number of datawords in the data FIFO is greater than the value in FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request will be generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, the value in the FTHLD field must be set to be less than or equal to the FIFO size as indicated by FSIZE.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 FSIZE	FIFO Size

Table continues on the next page...

SDADC_FCR field descriptions (continued)

Field	Description
	The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 dataword 01 FIFO depth = 4 datawords 10 FIFO depth = 8 datawords 11 FIFO depth = 16 datawords
31 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one dataword regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple datawords; FIFO depth is one dataword 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE

39.6.8 Software Trigger Key Register (SDADC_STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																ST_KEY															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_STKR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 ST_KEY	Software Trigger Key This bitfield, when written with FFFFh, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the chip's implementation. Any write access to this register which is different from the predefined key is ignored. Read access will always return 0000h.

39.6.9 Converted Data Register (SDADC_CDR)

The Converted Data Register (CDR) stores the converted data in the data FIFO.

The conversion result in the CDATA word is a 16-bit, signed value.

NOTE

When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CDATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_CDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CDATA	Converted Data Register (CDATA[13:0]) The converted datawords can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 15 down to 0) coming from ADC.

39.6.10 WDG Threshold Register (SDADC_WTHHLR)

This is the Converted Data Range WDG Threshold register.

NOTE

A single WDG monitor will check for lower and upper threshold crossover events.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	THRH																THRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_WTHHLR field descriptions

Field	Description
0–15 THRH	WDG Upper Threshold Value This indicates the upper threshold for the converted data on any channel, above which threshold a crossover event would be flagged through SFR[WTHH].
16–31 THRL	WDG Lower Threshold Value This indicates the lower threshold for the converted data on any channel, below which threshold cross over event would be flagged through SFR[WTHL].

39.7 Functional description

The SDADC has the following distinct available modes determined by fields in MCR:

- Differential input mode
- Single-ended input mode
- External modulator mode
- Low consumption mode

39.7.1 Differential input mode

Differential input mode is entered when the SDADC_MCR[MODE] field is set to '0' and the SDADC_MCR[EN] field is set to '1'. In differential input mode, a pair of analog inputs among AN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AN[0,1], AN[2,3], AN[4,5], and AN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the SDADC_CSR[ANCHSEL] field. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative terminal, is determined according to the Analog input AN[0:7] selection table in the Channel Selection Register description.

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

39.7.2 Single-ended input mode

Single-ended input mode is entered by setting the SDADC_MCR[MODE] and SDADC_MCR[EN] fields to '1'. In this mode, one of the analog inputs among AN[0:7] is selected via the SDADC_CSR[ANCHSEL] field and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled vua the SDADC_MCR[VCOMSEL] field.

39.7.3 External modulator mode

This mode is selected by asserting the MCR[EMSEL] field. When external modulator is selected, the internal modulator can be put in low consumption mode by deasserting the MCR[EN] field. In this mode, the data stream and clock outputs from the external modulator available on EMDATA/EMCLK input pins are multiplexed with internal modulator data output and input sampling clock (f_s). External modulator data must be synchronized with respect to the falling edge of EMCLK. In this mode, hardware trigger input is ignored even if it is enabled by asserting the MCR[TRIGEN] field. All other configuration settings (for example gain, decimation rate, filter, etc.) programmed in MCR have no effect on the conversion.

39.7.4 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

39.7.5 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to the Analog input AN[0:7] selection table in the Channel Selection Register description. Each analog input can be biased to half scale (AVDD/2) via on-chip 100 kΩ resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

39.7.6 Programmable gain and decimation rate

All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s. The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different passbands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

39.7.7 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The –3 dB frequency of the filter is fixed (not programmable) and is specified in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC and is applicable only to internal modulator mode.

39.7.8 Data conversion

The SDADC operates in a continuous conversion mode as soon as it is enabled. There is an inherent settling time required after SDADC startup before conversion data is considered valid.

NOTE

During that period, the Converted Data Valid Flag bit (SDADC_SFR[CDVF]) reads '0'.

Settling time duration depends on the latency and output settling time specifications. To account for the settling time, an internal timer is implemented that counts down from a start value determined by the value of the SDADC_OSDR[OSD] field.

The internal countdown timer is triggered by an SDADC reset, i.e., by writing the value 5AF0h to the SDADC_RKR[RESET_KEY] field.

NOTE

An SDADC reset is required any time there is a change to the SDADC configuration (via changes to the SDADC_MCR register settings), or a change to analog channel selection (via SDADC_CSR register settings).

When the timer reaches '0', the SDADC_SFR[CDVF] field is asserted and converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

Any write access to the SDADC_MCR or SDADC_CSR register causes the SDADC_SFR[CDVF] flag to be cleared, indicating that data coming from SDADC is not valid. If the SDADC_SFR[CDVF] flag is '0', the following flags are blocked from becoming asserted:

- Data FIFO Full Flag (SDADC_SFR[DFFF])
- Data FIFO Overrun Flag (SDADC_SFR[DFORF])

If the SDADC_SFR[DFFF] bit is already set, the data FIFO is already full before SDADC_SFR[CDVF] deassertion.

Wraparound Control Mechanism: Here a wraparound logic includes all analog channels of the input mutiplexer from Ain("000") to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by hardware or software source:

- MCR[WRMODE] bit would enable or disable the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- Wraparound Counter: An internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value "000". Maximum value of the counter (wraparound value) should be programmed in CSR[ANCHSEL_WRAP] before entering wraparound mode. The definition of ANCHSEL and ANCHSEL_WRAP is provided in the Analog input AN[0:7] selection table in the Channel Selection Register description.
- Upon entering or during wraparound mode user should take care to program ANCHSEL value less than ANCHSEL_WRAP value; otherwise hardware would reject the new ANCHSEL value.
- Possible trigger sources:
  - SOURCE1 (Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse. Each such trigger initiates a fresh conversion and advances the wraparound counter by one.

NOTE

Care should be taken while programming the TRIGEN/ TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.

  - SOURCE2 (Hardware Trigger): Here external trigger output enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger. Each such valid

trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.

- Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution. This can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value and the wraparound counter would increment from the updated value.
- Current wraparound counter value can be read from the SFR[ANCHSEL_CNT] bitfield.

39.7.9 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously for four cycles. This will ensure that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event will be ignored. A hardware trigger event is also ignored when external modulator is selected (MCR[EMSEL] = 1).

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time. Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSDR[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter will be less than 100 ns. It will be more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: In wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source should be selected before enabling the wraparound mode.

39.7.10 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted datawords stored in data FIFO is equal to or greater than the value indicated by threshold determined by FCR[FTHLD]. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted datawords cannot be received and data is lost. The interrupt request is generated only when RSER[DFORIE] is asserted.

Apart from above, a WDG threshold crossover event can also generate an interrupt or DMA request.

If RSER[GDIGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted.

39.7.11 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Enable gain error calibration mode by writing MCR[GECEN] to '1'.
3. Configure the mux selection ANCHSEL field of CSR to '110'. This configuration will apply VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 00h.
5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
7. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0111_1001_1001_1001b, corresponding to full positive scale after attenuation inserted by the internal filter (1*0.95). The measurement should be repeated to reduce contribution of noise during the calibration process. D_p is the average value of attenuated positive full scale given by D_p = AVERAGE(CDR[CDATA]).
8. Change the mux selection ANCHSEL field of CSR to '111'. This configuration will apply VREFN on positive terminal and VREFP on negative terminal.
9. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.

10. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 1000_0110_0110_0110b, corresponding to full negative scale after attenuation inserted by the internal filter (-1*0.95). The measurement should be repeated to reduce contribution of noise during calibration process. D_n is the average value of attenuated negative full scale given by D_n = AVERAGE(CDR[CDATA]).
11. The SDADC calibrated gain can be calculated as:
    $Gain = (D_p - D_n) / 2^{16}$
12. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC should be normalized using the calculated gain: $CDATA_{norm}$ is given by $CDATA_{norm}$ = CDR[CDATA]/Gain.

Note

a. The gain error calibration should only be done with gain equal to '1' (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN>1.
b. MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit should be set both during calibration process and after calibration, for application data conversion. Application data conversion must then be normalized using the measured gain.
c. The higher the number of full-scale conversion (D_p, D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

39.7.12 Offset calibration support

To perform offset calibration, the following sequence must be applied.

1. Select differential mode of operation by writing SDADC_MCR[MODE] to '0'.
2. Configure the SDADC_CSR[ANCHSEL] mux selection field to '100' or '101' as required.
   a. SDADC_CSR[ANCHSEL] ='100' in case of data conversion after calibration in "single ended mode with negative input=$V_{SS\_HV\_ADR\_D}$"
   b. SDADC_CSR[ANCHSEL] ='101' in case of data conversion after calibration in "differential mode" and "single ended mode with negative input = $(V_{DD\_HV\_ADR\_D} - V_{SS\_HV\_ADR\_D}) / 2$"

3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 00h.
4. Disable the high pass filter by deasserting SDADC_MCR[HPFEN].
5. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output. Expected output is 00_0000_0000_0000b. The SDADC offset can be calculated as:
   Offset = Expected Output – Actual Output

Note

The offset must be calculated for each SDADC_MCR[PGAN] field setting since it is expected to vary with gain configuration of SDADC.

39.8 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter if required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC startup time or latency from reset exit.
6. Generate a reset event by writing 5AF0h to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].
7. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

The following sequence needs to be followed when external modulator mode of SDADC is selected for data conversions.

1. Disable the SDADC internal modulator by deasserting MCR[EN].
2. Configure MCR to select the external modulator mode.
3. Select the required external pins EMDATA/EMCLK in the system integration logic (SIUL) of the device.
4. Configure OSD delay according to external modulator startup time or latency from reset exit.
5. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.

6. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

39.8.1 Data conversion step

To acquire a data from SDADC, the following sequence is required:

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter is required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC required startup time or latency from reset exit.
6. Generate a reset event writing 5AF0h in RESET_KEY field of RKR register.
7. Wait till FIFO empty flag DFEF of SFR register is clear.
8. Read data by CDATA of CDR register.

Repeat steps from 6 to 8 for new acquisitions.

NOTE

The time elapsed between reset event and Read data by CDATA field of CDR register are:

- Reset event <- -> Data Valid flag(CDVS) = Output settling Time delay (OSDR has to be set at least to 16) (Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.).
- Data Valid flag <- -> Data FIFO is not empty = 0.5*CLK_Out + 3*CLK_In
- Data FIFO is not empty <- -> read Data (safety point) = 2*CLK_In, where CLK_Out = CLK_In/(2*OSR)

Chapter 40
Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

40.1 Introduction

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for the application. It provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has its corresponding digital interface implemented at the chip level.

40.2 Overview

The SARADC digital interface contains advanced features for normal or injected conversion modes of operation. The conversion can be triggered by software or hardware.

The digital interface can be configured to control up to 256 multiplexed analog input channels. There are three types of input channels:

- Internal channels, mapped to index ranging from 0 to 95
- Test channels, mapped to index ranging from 96 to 127
- External channels, mapped to index ranging from 128 to 255

Note

The number of input channels and their mapping varies depending on your device configuration. See the device-specific Analog-to-Digital Converters (ADC)

Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

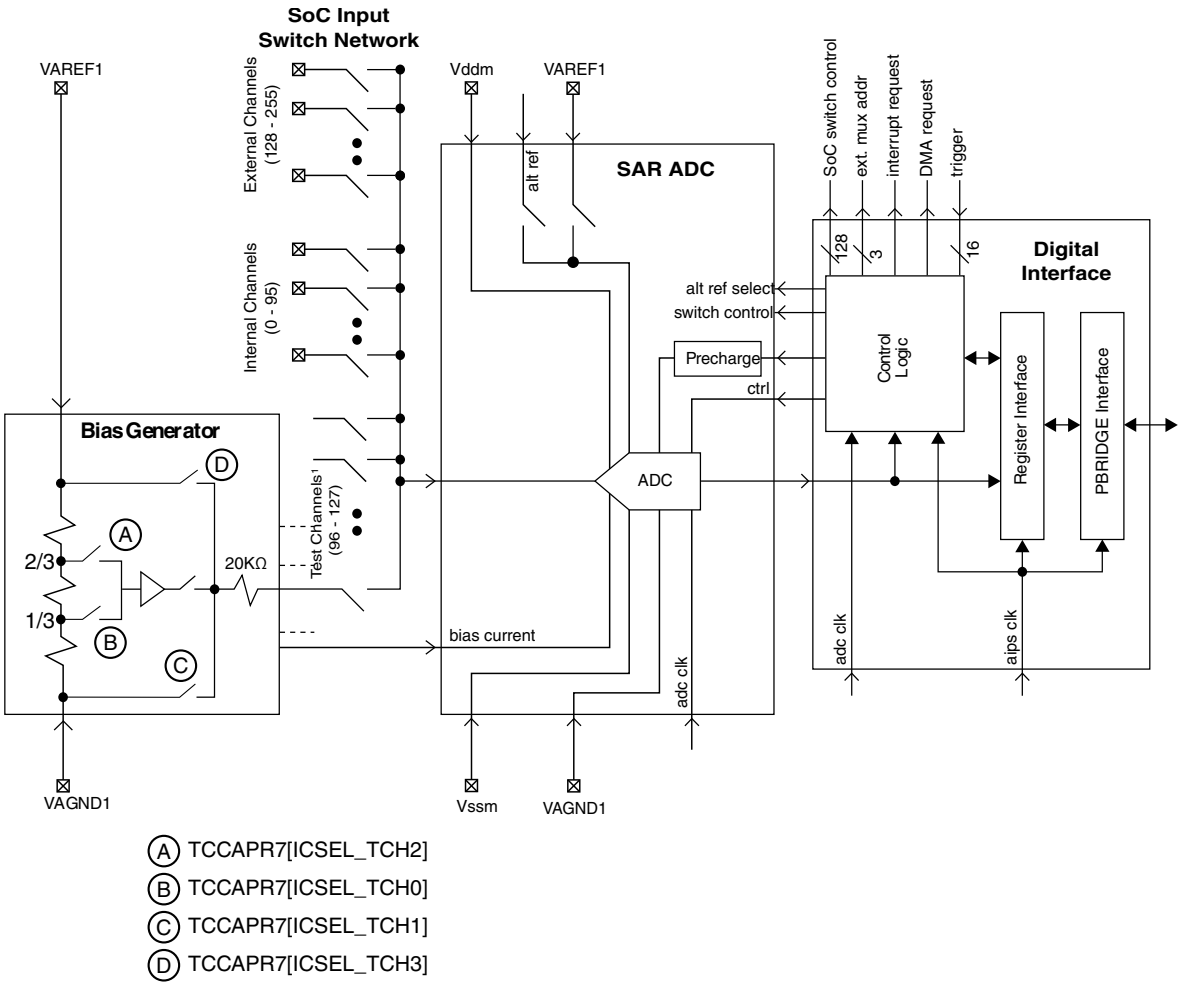
The mask registers present within the digital interface can be properly programmed to configure which channel has to be converted. External channel selection is provided through external decode signals and are available as alternate functions on GPIO. Each set of eight external channels can be mapped to any internal analog channel by static programming by software in specific registers.

Four conversion timing registers exist that allow configuration of different precharging and sampling durations, and it is possible to select one of the conversion timing registers for each channel. For each channel, it is possible to choose the voltage reference through software. Analog Watchdogs allow continuous hardware monitoring of analog input channels. The digital interface also provides for interrupt/DMA support for various conditions related to end of channel conversions.

The following figure shows the SARADC block diagram.

Note

SARADC_B implements a bias generator converting the Vddm, 1/3 Vddm, 2/3 Vddm, and Vssm reference points through the internal 20 kΩ source impedance.



[1] Test channels are mapped to internal analog channels via the SARADCB_TCCAPRx registers

Figure 40-1. SARADC block diagram

40.3 Feature description

- Selectable 10-bit or 12-bit data resolution output
- Up to 96 internal channels, 32 test channels, 128 external channels supported; variable number of analog channels of each type controlled by parameters
- 4 different conversion timing registers selectable for any channel
- Mapping of external channel to any internal channel through static programming by software
- Shorting of test channel with internal channel through static programming by software
- External decode signals (3 signals) for selection of external analog mux inputs

- Normal conversion with One Shot/Scan modes
- Injected conversion with One Shot mode
- Normal conversion with dedicated trigger input
- Injected conversion with dedicated trigger input
- 2 different abort features that allow to abort either a single channel conversion or chain conversion
- Reference selection for each channel
- Power Down Mode
- Dedicated data register for each channel, containing the following information:
  - 10-bit or 12-bit conversion result in one half-word
  - Status byte which provides some conversion information such as mode of operation (Normal, Injected), data valid, data overwritten status.
  - Control byte for reference selection, conversion timing parameter selection
- Configurable number of analog watchdogs
  - Trigger outputs on watchdog threshold crossover events
- Interrupt/DMA support for the following conditions
  - End of conversion of single channel for both normal, injected conversions
  - End of conversion chain for both normal, injected conversions
  - Watchdog thresholds crossover

40.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

40.4.1 Normal channel conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (ICNCMR0-2, TCNCMR, ECNCMR0-3). Each channel can be individually enabled by setting '1' in the corresponding field of these registers. Mask registers must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (MSR[NSTART] bit is reset).

40.4.1.1 Start of normal conversion

The normal conversion can be started as follows:

- By software
  - The normal conversion chain starts when the MCR[NSTART] bit is set. It is recommended that the normal trigger enable MCR[NTRGEN] bit is reset during conversions started by software. Once a normal conversion has started, any further write operation of the MCR[NSTART] bit will have no effect during ongoing conversion, but the conversion will be enqueued after completion of current conversion in oneshot mode.
- By normal trigger
  - A normal trigger valid edge or level is detected to start the conversion. The normal trigger is enabled by setting the NTRGEN bit in MCR and two options are available. A programmed event (rising, falling or both edges depending on NEDGESEL bitfield of MCR) on the normal trigger input starts the normal conversion. Once normal conversion has started, any further valid trigger edge will be ignored during ongoing conversion. If the trigger mode is enabled (MCR[NTRGEN] bit is set), then software can only clear the MCR[NSTART] bit for terminating the conversion process. It is not possible to software to give any conversion request.

NOTE

An appropriate gap should be maintained between any two consecutive triggers such that next trigger is received only after the completion of ongoing conversion. This gap value can calculated as a sum of total conversion time as programmed using the CTR0-3 registers for each channel in the chain.

40.4.1.2 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot Mode
- Scan Mode

To enter one of these modes, it is necessary to program the MCR[MODE] bit. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in the following figure.

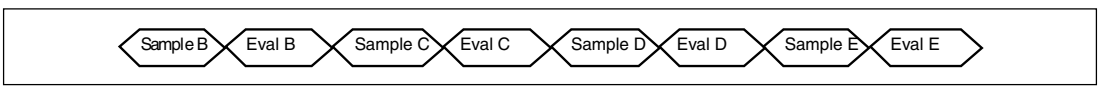


Figure 40-2. Normal conversion flow

In **One Shot Mode** (MODE = 0) a sequential conversion specified in the ICNCMR0-2, TCNCMR, ECNCMR0-3 mask registers is performed only once. At the end of each conversion, the digital result of the conversion is stored into the corresponding data register.

For Example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *One Shot Mode*. MODE = 0 is set for One Shot mode. Conversion starts from the channel B followed by conversion of channels D-E. At the end of conversion of channel E the scanning of channels stops.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time MCR[NSTART] bit is reset if the conversion is started by software.

Once the normal conversion operation starts in oneshot mode and another software conversion request is made writing the MCR[NSTART] bit and before MSR[NSTART] is cleared, one additional conversion is initiated with the same channel mask register settings after the current conversion is finished, this is called **enqueuing**. Enqueuing can help in removing the delay between two consecutive oneshot chains and uncertainty caused by metastability between MCR[NSTART] bit written at system clock and generation of start pulse on ADC clock before every chain execution.

In **Scan Mode** (MODE = 1), a sequential conversion of N channels specified in the ICNCMR0-2, TCNCMR, ECNCMR0-3 registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

In this mode, the software can start conversion by writing '1' to MCR[NSTART] bit or a valid normal trigger edge is detected which also sets MCR[NSTART] bit. The MCR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One Shot Mode, the MCR[NSTART] bit is not reset. It can be reset by software when the user needs to stop scan mode. In that case, the current scan conversion is completed and MCR[NSTART] bit also is reset after the last conversion of the chain.

For Example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself till the MCR[NSTART] bit is reset by software.

In both the modes, at the end of each channel conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit) and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit). The End Of Chain interrupt is issued for each pass of scan mode.

40.4.2 Injected channel conversion

A conversion chain can be injected into the ongoing normal conversion configuring the injected mask registers (ICJCMR0-2, TCJCMR, ECJCMR0-3). As normal conversion, each internal, test or external channel can be individually selected. This injected conversion can only be in one-shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped as shown in the following figure.

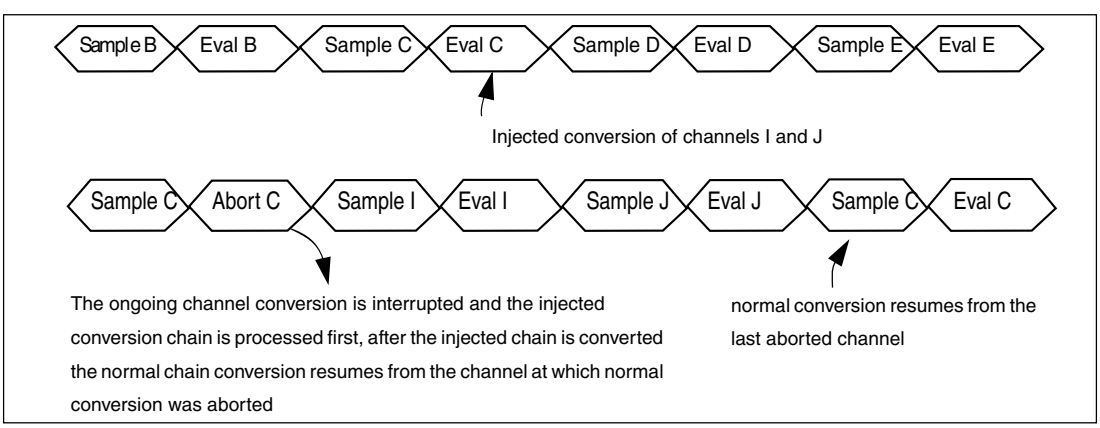


Figure 40-3. Injected sample/conversion sequence

The injected conversion can be started as follows:

- By software
  - The injected conversion chain starts when the MCR[JSTART] bit is set. The current conversion is suspended and the injected chain is converted. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed. It is recommended that the injection trigger enable MCR[JTRGEN] bit is reset during conversions started by software. Once injected conversion has started, any further write operation of MCR[JSTART] bit will not have any effect during ongoing conversion.
- By injection trigger
  - The injection trigger is enabled by setting the JTRGEN bit in MCR. A programmed event (rising, falling or both edges depending on JEDGESEL bitfield of MCR) on the injection external trigger starts the injected conversion and also sets the MSR[JSTART] bit. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed.
  - When JTRGSEQ bit is set, the injection trigger sequence feature is enabled. In this mode, each valid edge of injection trigger input will convert one analog channel. First valid edge will convert the first analog channel enabled through ICJCMR0-2, TCJCMR, ECJCMR0-3 registers. The next valid edge converts the next enabled analog input and so on. Once all the enabled channels are converted, the channel selection loops back to the first enabled channel for next trigger edge. If the JTRGSEQ is reset, the channel selection is reset again to first channel enabled through ICJCMR0-2, TCJCMR, ECJCMR0-3 registers. If JTRGSEQ is reset before completing the injected chain, the remaining channels are completed in one-shot after receiving next valid trigger edge.

NOTE

An appropriate gap should be maintained between any two consecutive injected triggers such that next injected trigger is received only after the completion of ongoing injected conversion. This gap value can calculated as a sum of total conversion time as programmed using CTR0-3 registers for each channel in the injected chain.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time MSR[JSTART] is reset if the conversion is started by software.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

40.4.3 Abort conversion

Two different abort functions are provided.

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the SARADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported generating an ECH interrupt.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to '1', a new chain conversion is started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

  When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended Normal conversion, both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset).

40.4.4 Analog conversion timings and reference selection

- Conversion timings
- Alternate reference selection

40.4.4.1 Conversion timings

In order to support different loadings and switching times (in particular for the external channel types) four different Conversion Timing registers are present (CTR0-3). Each conversion timing register contains PRECHG, INPSAMP bitfields to program the required duration for precharging and sampling phases. The selection of these registers for each channel is done by the SAMPSEL bitfield of corresponding channel data register.

Precharging phase allows to precharge or discharge the internal capacitor to a fixed internal voltage for duration determined by PRECHG bitfield prior to actual analog channel sampling start. This is useful for resetting information regarding the last converted data. Precharging feature can be enabled or disabled for each channel by programming the PCE bit of corresponding channel data register. If precharging is enabled for a given channel, the normal sequence of operation will be precharging, sampling followed by evaluation as shown in the following figure. Please refer to device datasheet for the minimum precharge and sample durations required.

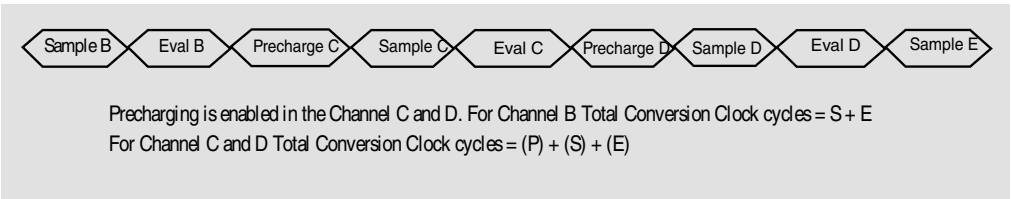


Figure 40-4. Analog conversion sequence

Bitfields PRECHG, INPSAMP are used to define the total conversion duration (t_{conv}) and in particular the partition among precharge duration (t_{prechg}), sampling phase duration (t_{sample}) and evaluation phase duration (t_{eval}).

The precharging phase duration is given by

$$t_{prechg} = PRECHG * t_{ck}$$

where PRECHG must be greater than or equal to 2 (hardware requirement) and PCE bit of channel data register is '1'. In case the value of PRECHG is found to be less than 2, it is automatically set to 2 inside SARADC. If the PCE bit is '0', the precharging phase is skipped and conversion starts with sampling phase directly.

The sampling phase duration is given by the following equation:

$$t_{sample} = INPSAMP * t_{ck}$$

where INPSAMP must be greater than or equal to 4 (hardware requirement). In case the value of INPSAMP is found to be less than 5, it is automatically set to 5 inside SARADC.

The total evaluation phase duration is given by the following equations:

$$t_{eval} = 25 * t_{ck}$$
$$t_{eval} = 21 * t_{ck} \text{ (for 10-bit conversion)}$$

In this phase, the internal sampling capacitor is disconnected from the analog channel and ADC estimates the digitized value of the sampled input using successive approximation algorithm. In the evaluation phase, all the bits are estimated sequentially to provide the conversion result.

The total conversion duration is (not including external multiplexing) is given by the following:

$t_{conv} = t_{prechg} + t_{sample} + t_{eval}$

The timings refer to the unit t_{ck} refers to reciprocal of f_{ck}, where f_{ck} = SARADC peripheral clock. Total conversion time is < 5µs.

Note

The current implementation of ADCDIG does not support hardware restriction over values of PRECHG and INPSAMP fields. As minimum values for these fields are product specific and need to be managed through software.

40.4.4.2 Alternate reference selection

In order to support different accuracy requirements, a voltage reference can be selected for each channel through REFSEL bit of the corresponding channel data register (for example, ICDR*n*). For some channels, this bit REFSEL is not software programmable and the selection configuration is fixed based on device-specific requirements.

40.4.5 Test channel connection with internal analog channel

The test channels are meant for monitoring various on-chip analog signals coming from temperature sensor, bandgap, voltage regulator and different voltage levels available on the chip. These analog signals can be interfaced to SARADC through test channels for testing purpose. Test channels are mapped from 96 to 127. Each test channel is controlled in the same way as normal internal channel.

SAR ADC Self Testing method: The test channel [127:124] is also mapped to various V_ref voltages (4 voltages) selection switches in the ADCBIAS block, the decoding of these channels to switch on and select a particular V_ref voltage in done in SoC level muxing logic. Refer to the ADC configuration for detailed mapping/control description of the Self Test feature.

It is also possible to perform test channel conversions with and without shorting the normal analog channel pin with the internal test voltage. The test channel to internal channel mapping is enabled through top level synthesis parameter testch_short_intch vector, if shorting of a particular test channel has to be disabled then parameter should be set to ‘0’ for corresponding test channel. This will result in removal of corresponding TCCAPR registers. For implemented TCCAPR registers to enable the short between test

channel and any internal analog channel, ESIC_TCHx bit of TCCAPR0-7 registers should be set. The mapping of test channel to any internal analog channel is determined by the ICSEL_TCHx bitfield of TCCAPR0-7 registers.

For analog test channel assignment, refer to the device configuration or ADC configuration section.

40.4.6 External channel mapping to internal analog channel

External channels are supposed to be used in conjunction with a given internal analog channel along with an 8:1 external mux outside the device. Each input of the external mux can be mapped to a unique channel number, thus allowing it to perform conversions like any other channel of the ADC. External channels are mapped from 128 to 255.

External channels are implemented in the multiples of 8. Each slice of eight channels can be associated to any physical internal analog channel mapped from 0 to 95. This external-to-internal channel mapping is determined by the ICSEL_ECx_x bitfield of ECMICR0-3 registers. For example, when external multiplexer is planned to be used on internal channel 5, and the external inputs need to be mapped to channels 128 to 135, then this channel number 5 must be written to the ICSEL_ECH128_135 bitfield of the ECMICR0 register. After reset, all external channels are mapped to internal analog channel 0.

40.4.7 Programmable analog watchdog

The analog watchdogs are used for determining whether the result of the conversion of a channel lies within a given guard area (as shown in the the following figure) specified by an upper and a lower threshold value named THRH and THRL respectively.

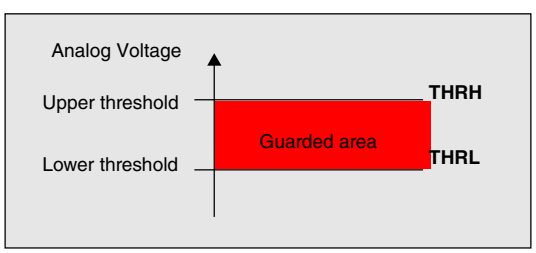


Figure 40-5. Analog watchdog guarded area

After the conversion of the selected channel a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guard area then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guard area, then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in the following table. Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 40-1. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

The lower and higher threshold values for the analog watchdog are programmed using THRHLR0-15 registers. Analog watchdog functionality for each channel can be enabled independently from ICWENR0-2, TCWENR, ECWEN0-3 registers and can select the watchdog threshold registers (THRHLR0-15) to be used by programming ICWSELR0-11, TCWSELR0-3, ECWSELR0..15 registers. The threshold registers selected by WSEL_CHx field of watchdog threshold selection registers will provide the threshold values.

For example, if channel 30 is to be monitored with the threshold values in register THRHLR12, then WSEL_CH30 is programmed to select THRHLR12 register to provide threshold values. The enabling is done by setting the bit corresponding to channel 30 in ICWENR0 register.

If the converted value for a particular channel lies outside the range specified by threshold values, then corresponding bit is set in ICAWORR0-2, TCAWORR, ECAWORR0-3 registers. In this case, a set of threshold registers (THRHLR0-15) can be linked to several analog channels. The threshold values to be selected for a channel needs be programmed only once in ICWSELR, TCWSELR, ECWSELR registers.

There is an external trigger output port which provides a single clock cycle pulse to indicate WDG threshold crossover event on a particular WDG (up to 16 trigger outputs for upto 16 WDG instances).

Note

If the higher threshold for the analog watchdog is programmed lower than the lower threshold and the converted value is lesser than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the

higher threshold) then the interrupt WDGxH for high threshold violation is set. Thus the user should take care of avoiding that situation as it could lead to misinterpretation of the watchdog interrupts.

40.4.8 DMA functionality

A Direct Memory Access (DMA) request can be programmed after the conversion of every channel, by setting the respective masking bit in ICDSR0-2, TCDSR and ECDSR0-3 registers. The DMA masking registers must be programmed before starting any conversion. The DMA transfers can be enabled using the DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, then the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

40.4.9 Interrupts

The SARADC digital interface generates the following maskable interrupt signals:

- End Of Conversion interrupts
  - NEOC (end of normal conversion of each channel)
  - NECH (end of normal chain)
  - JEOC (end of injected conversion of each channel)
  - JECH (end of injected chain)

The EOC_CH[x] bit of interrupt pending registers (ICIPR0-2, TCIPR, ECIPR0-3) is set when channel x completes the conversion (normal or injection). This pending status is qualified when the corresponding mask bit IM_CH[x] is also set in the interrupt mask registers (ICIMR0-2, TCIMR, ECIMR0-3).

The NEOC interrupt is generated only if the following conditions are met

- ISR[NEOC] bit is set after the normal conversion completion for any channel x.
- IMR[MSKNEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The JEOC interrupt is generated only if the following conditions are met

- ISR[JEOC] bit is set after the injected conversion completion for any channel x.
- IMR[MSKJEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

It is recommended to clear the NEOC/JEOC/EOCTU bits of ISR and all the individual channel pending bits which are not masked while servicing the interrupt.

The ISR[NECH] bit is set when a normal channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

The ISR[JECH] bit is set when an injected channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

- WDGxL and WDGxH (Watchdog threshold) interrupt requests

The interrupts generated due to the analog watchdog are handled by registers WTISR and WTIMR in order to check and enable the interrupt requests to external interrupt controller module. The watchdog interrupt source sets two pending bits WDGxH and WDGxL for each channel being monitored in the WTISR. The interrupt request is generated if at least one of the following conditions are met

- WTISR[WDGxL] bit is set and the corresponding mask bit MSKWDGxL is also set in WTIMR for at least one channel.
- WTISR[WDGxH] bit is set and the corresponding mask bit MSKWDGxH is also set in WTIMR for at least one channel.

The ISR and WTISR contain the interrupt pending request status flags. In case the user wants to clear a particular interrupt event status, writing a '1' to the corresponding status bit will clear the pending interrupt flag. (At this write operation, all other bits of the ISR must be maintained at '0'.)

The ADC module provides dedicated interrupt outputs for all interrupt sources. Also, all interrupt lines are OR-ed together to provide a single output to the external interrupt controller if a combined interrupt is required.

40.4.10 External decode signals selection and delay

The ADC provides three external decode signals used as select lines for 8:1 external mux. For each set of 8 external channels (for example [CH128:CH135], [CH136:CH143], and so on), the same 3-bit decode signals output will be used to select the inputs of external mux. These external mux select lines change in grey code sequence for each increment of channel selection. This ensures that only one line changes and avoids any current spikes or leakage if more than one line changes simultaneously.

In order to take into account the control switching time of the external analog mux, a Decode Signals Delay register is provided. Writing that register allows programming of the delay between the decoding signal selection and the actual start of conversion.

40.4.11 Power down mode

The SARADC analog block can be put in low current consumption mode by setting the PWDN bit in the MCR. This is the default mode after exiting from reset. This state must be exited before starting any operation by resetting the PWDN bit in the MCR. When in powerdown mode, no conversion can be started. Bit ADCSTATUS[0] in the MSR is set only when ADC enters Powerdown mode.

The power-down mode can be requested at any time by setting the PWDN bit in the MCR. If a conversion is ongoing, the SARADC cannot immediately enter the power-down mode. In fact, the SARADC enters power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit.

After the power down phase is completed the process ongoing before the power down phase must be restarted manually (by setting the appropriate START bit). Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

40.5 Memory map and register definition

This section provides memory map for the SARADC digital interface.

This section provides memory map for the SARADC digital interface.

SARADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Main Configuration Register (SARADC_MCR)	32	R/W	0000_8001h	40.5.1/1618
4	Main Status Register (SARADC_MSR)	32	R	0000_0001h	40.5.2/1621
10	Interrupt Status Register (SARADC_ISR)	32	w1c	0000_0000h	40.5.3/1623
14	Internal channel Interrupt Pending Register (SARADC_ICIPR0)	32	w1c	0000_0000h	40.5.4/1624
18	Internal channel Interrupt Pending Register (SARADC_ICIPR1)	32	w1c	0000_0000h	40.5.4/1624
1C	Internal channel Interrupt Pending Register (SARADC_ICIPR2)	32	w1c	0000_0000h	40.5.4/1624
20	Interrupt Mask Register (SARADC_IMR)	32	R/W	0000_0000h	40.5.5/1625
24	Internal Channel Interrupt Mask Register (SARADC_ICIMR0)	32	R/W	0000_0000h	40.5.6/1626
28	Internal Channel Interrupt Mask Register (SARADC_ICIMR1)	32	R/W	0000_0000h	40.5.6/1626
2C	Internal Channel Interrupt Mask Register (SARADC_ICIMR2)	32	R/W	0000_0000h	40.5.6/1626
30	Watchdog Threshold Interrupt Status Register (SARADC_WTISR)	32	w1c	0000_0000h	40.5.7/1626
34	Watchdog Threshold Interrupt Mask Register (SARADC_WTIMR)	32	R/W	0000_0000h	40.5.8/1630
40	DMA Enable Register (SARADC_DMAE)	32	R/W	0000_0000h	40.5.9/1634
44	Internal Channel DMA Select Register (SARADC_ICDSR0)	32	R/W	0000_0000h	40.5.10/ 1635
48	Internal Channel DMA Select Register (SARADC_ICDSR1)	32	R/W	0000_0000h	40.5.10/ 1635
4C	Internal Channel DMA Select Register (SARADC_ICDSR2)	32	R/W	0000_0000h	40.5.10/ 1635
60	Watchdog Threshold Register (SARADC_WTHRHLR0)	32	R/W	0FFF_0000h	40.5.11/ 1636
64	Watchdog Threshold Register (SARADC_WTHRHLR1)	32	R/W	0FFF_0000h	40.5.11/ 1636
68	Watchdog Threshold Register (SARADC_WTHRHLR2)	32	R/W	0FFF_0000h	40.5.11/ 1636
6C	Watchdog Threshold Register (SARADC_WTHRHLR3)	32	R/W	0FFF_0000h	40.5.11/ 1636
94	Conversion Timing Register (SARADC_CTR0)	32	R/W	0000_0000h	40.5.12/ 1636
98	Conversion Timing Register (SARADC_CTR1)	32	R/W	0000_0000h	40.5.12/ 1636
9C	Conversion Timing Register (SARADC_CTR2)	32	R/W	0000_0000h	40.5.12/ 1636
A0	Conversion Timing Register (SARADC_CTR3)	32	R/W	0000_0000h	40.5.12/ 1636

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Internal Channel Normal Conversion Mask Register (SARADC_ICNCMR0)	32	R/W	0000_0000h	40.5.13/ 1637
A8	Internal Channel Normal Conversion Mask Register (SARADC_ICNCMR1)	32	R/W	0000_0000h	40.5.13/ 1637
AC	Internal Channel Normal Conversion Mask Register (SARADC_ICNCMR2)	32	R/W	0000_0000h	40.5.13/ 1637
B4	Internal Channel Injected Conversion Mask Register (SARADC_ICJCMR0)	32	R/W	0000_0000h	40.5.14/ 1638
B8	Internal Channel Injected Conversion Mask Register (SARADC_ICJCMR1)	32	R/W	0000_0000h	40.5.14/ 1638
BC	Internal Channel Injected Conversion Mask Register (SARADC_ICJCMR2)	32	R/W	0000_0000h	40.5.14/ 1638
C8	Power Down Exit Delay Register (SARADC_PDEDR)	32	R/W	0000_0000h	40.5.15/ 1638
100	Internal Channel Data Register (SARADC_ICDR0)	32	R/W	0000_0000h	40.5.16/ 1639
104	Internal Channel Data Register (SARADC_ICDR1)	32	R/W	0000_0000h	40.5.16/ 1639
108	Internal Channel Data Register (SARADC_ICDR2)	32	R/W	0000_0000h	40.5.16/ 1639
10C	Internal Channel Data Register (SARADC_ICDR3)	32	R/W	0000_0000h	40.5.16/ 1639
110	Internal Channel Data Register (SARADC_ICDR4)	32	R/W	0000_0000h	40.5.16/ 1639
114	Internal Channel Data Register (SARADC_ICDR5)	32	R/W	0000_0000h	40.5.16/ 1639
118	Internal Channel Data Register (SARADC_ICDR6)	32	R/W	0000_0000h	40.5.16/ 1639
11C	Internal Channel Data Register (SARADC_ICDR7)	32	R/W	0000_0000h	40.5.16/ 1639
120	Internal Channel Data Register (SARADC_ICDR8)	32	R/W	0000_0000h	40.5.16/ 1639
124	Internal Channel Data Register (SARADC_ICDR9)	32	R/W	0000_0000h	40.5.16/ 1639
128	Internal Channel Data Register (SARADC_ICDR10)	32	R/W	0000_0000h	40.5.16/ 1639
12C	Internal Channel Data Register (SARADC_ICDR11)	32	R/W	0000_0000h	40.5.16/ 1639
130	Internal Channel Data Register (SARADC_ICDR12)	32	R/W	0000_0000h	40.5.16/ 1639
134	Internal Channel Data Register (SARADC_ICDR13)	32	R/W	0000_0000h	40.5.16/ 1639
138	Internal Channel Data Register (SARADC_ICDR14)	32	R/W	0000_0000h	40.5.16/ 1639

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13C	Internal Channel Data Register (SARADC_ICDR15)	32	R/W	0000_0000h	40.5.16/ 1639
140	Internal Channel Data Register (SARADC_ICDR16)	32	R/W	0000_0000h	40.5.16/ 1639
144	Internal Channel Data Register (SARADC_ICDR17)	32	R/W	0000_0000h	40.5.16/ 1639
148	Internal Channel Data Register (SARADC_ICDR18)	32	R/W	0000_0000h	40.5.16/ 1639
14C	Internal Channel Data Register (SARADC_ICDR19)	32	R/W	0000_0000h	40.5.16/ 1639
150	Internal Channel Data Register (SARADC_ICDR20)	32	R/W	0000_0000h	40.5.16/ 1639
154	Internal Channel Data Register (SARADC_ICDR21)	32	R/W	0000_0000h	40.5.16/ 1639
158	Internal Channel Data Register (SARADC_ICDR22)	32	R/W	0000_0000h	40.5.16/ 1639
15C	Internal Channel Data Register (SARADC_ICDR23)	32	R/W	0000_0000h	40.5.16/ 1639
160	Internal Channel Data Register (SARADC_ICDR24)	32	R/W	0000_0000h	40.5.16/ 1639
164	Internal Channel Data Register (SARADC_ICDR25)	32	R/W	0000_0000h	40.5.16/ 1639
168	Internal Channel Data Register (SARADC_ICDR26)	32	R/W	0000_0000h	40.5.16/ 1639
16C	Internal Channel Data Register (SARADC_ICDR27)	32	R/W	0000_0000h	40.5.16/ 1639
170	Internal Channel Data Register (SARADC_ICDR28)	32	R/W	0000_0000h	40.5.16/ 1639
174	Internal Channel Data Register (SARADC_ICDR29)	32	R/W	0000_0000h	40.5.16/ 1639
178	Internal Channel Data Register (SARADC_ICDR30)	32	R/W	0000_0000h	40.5.16/ 1639
17C	Internal Channel Data Register (SARADC_ICDR31)	32	R/W	0000_0000h	40.5.16/ 1639
180	Internal Channel Data Register (SARADC_ICDR32)	32	R/W	0000_0000h	40.5.16/ 1639
184	Internal Channel Data Register (SARADC_ICDR33)	32	R/W	0000_0000h	40.5.16/ 1639
188	Internal Channel Data Register (SARADC_ICDR34)	32	R/W	0000_0000h	40.5.16/ 1639
18C	Internal Channel Data Register (SARADC_ICDR35)	32	R/W	0000_0000h	40.5.16/ 1639
190	Internal Channel Data Register (SARADC_ICDR36)	32	R/W	0000_0000h	40.5.16/ 1639

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
194	Internal Channel Data Register (SARADC_ICDR37)	32	R/W	0000_0000h	40.5.16/ 1639
198	Internal Channel Data Register (SARADC_ICDR38)	32	R/W	0000_0000h	40.5.16/ 1639
19C	Internal Channel Data Register (SARADC_ICDR39)	32	R/W	0000_0000h	40.5.16/ 1639
1A0	Internal Channel Data Register (SARADC_ICDR40)	32	R/W	0000_0000h	40.5.16/ 1639
1A4	Internal Channel Data Register (SARADC_ICDR41)	32	R/W	0000_0000h	40.5.16/ 1639
1A8	Internal Channel Data Register (SARADC_ICDR42)	32	R/W	0000_0000h	40.5.16/ 1639
1AC	Internal Channel Data Register (SARADC_ICDR43)	32	R/W	0000_0000h	40.5.16/ 1639
1B0	Internal Channel Data Register (SARADC_ICDR44)	32	R/W	0000_0000h	40.5.16/ 1639
1B4	Internal Channel Data Register (SARADC_ICDR45)	32	R/W	0000_0000h	40.5.16/ 1639
1B8	Internal Channel Data Register (SARADC_ICDR46)	32	R/W	0000_0000h	40.5.16/ 1639
1BC	Internal Channel Data Register (SARADC_ICDR47)	32	R/W	0000_0000h	40.5.16/ 1639
1C0	Internal Channel Data Register (SARADC_ICDR48)	32	R/W	0000_0000h	40.5.16/ 1639
1C4	Internal Channel Data Register (SARADC_ICDR49)	32	R/W	0000_0000h	40.5.16/ 1639
1C8	Internal Channel Data Register (SARADC_ICDR50)	32	R/W	0000_0000h	40.5.16/ 1639
1CC	Internal Channel Data Register (SARADC_ICDR51)	32	R/W	0000_0000h	40.5.16/ 1639
1D0	Internal Channel Data Register (SARADC_ICDR52)	32	R/W	0000_0000h	40.5.16/ 1639
1D4	Internal Channel Data Register (SARADC_ICDR53)	32	R/W	0000_0000h	40.5.16/ 1639
1D8	Internal Channel Data Register (SARADC_ICDR54)	32	R/W	0000_0000h	40.5.16/ 1639
1DC	Internal Channel Data Register (SARADC_ICDR55)	32	R/W	0000_0000h	40.5.16/ 1639
1E0	Internal Channel Data Register (SARADC_ICDR56)	32	R/W	0000_0000h	40.5.16/ 1639
1E4	Internal Channel Data Register (SARADC_ICDR57)	32	R/W	0000_0000h	40.5.16/ 1639
1E8	Internal Channel Data Register (SARADC_ICDR58)	32	R/W	0000_0000h	40.5.16/ 1639

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1EC	Internal Channel Data Register (SARADC_ICDR59)	32	R/W	0000_0000h	40.5.16/ 1639
1F0	Internal Channel Data Register (SARADC_ICDR60)	32	R/W	0000_0000h	40.5.16/ 1639
1F4	Internal Channel Data Register (SARADC_ICDR61)	32	R/W	0000_0000h	40.5.16/ 1639
1F8	Internal Channel Data Register (SARADC_ICDR62)	32	R/W	0000_0000h	40.5.16/ 1639
1FC	Internal Channel Data Register (SARADC_ICDR63)	32	R/W	0000_0000h	40.5.16/ 1639
200	Internal Channel Data Register (SARADC_ICDR64)	32	R/W	0000_0000h	40.5.16/ 1639
204	Internal Channel Data Register (SARADC_ICDR65)	32	R/W	0000_0000h	40.5.16/ 1639
208	Internal Channel Data Register (SARADC_ICDR66)	32	R/W	0000_0000h	40.5.16/ 1639
20C	Internal Channel Data Register (SARADC_ICDR67)	32	R/W	0000_0000h	40.5.16/ 1639
210	Internal Channel Data Register (SARADC_ICDR68)	32	R/W	0000_0000h	40.5.16/ 1639
214	Internal Channel Data Register (SARADC_ICDR69)	32	R/W	0000_0000h	40.5.16/ 1639
218	Internal Channel Data Register (SARADC_ICDR70)	32	R/W	0000_0000h	40.5.16/ 1639
21C	Internal Channel Data Register (SARADC_ICDR71)	32	R/W	0000_0000h	40.5.16/ 1639
220	Internal Channel Data Register (SARADC_ICDR72)	32	R/W	0000_0000h	40.5.16/ 1639
224	Internal Channel Data Register (SARADC_ICDR73)	32	R/W	0000_0000h	40.5.16/ 1639
228	Internal Channel Data Register (SARADC_ICDR74)	32	R/W	0000_0000h	40.5.16/ 1639
22C	Internal Channel Data Register (SARADC_ICDR75)	32	R/W	0000_0000h	40.5.16/ 1639
230	Internal Channel Data Register (SARADC_ICDR76)	32	R/W	0000_0000h	40.5.16/ 1639
234	Internal Channel Data Register (SARADC_ICDR77)	32	R/W	0000_0000h	40.5.16/ 1639
238	Internal Channel Data Register (SARADC_ICDR78)	32	R/W	0000_0000h	40.5.16/ 1639
23C	Internal Channel Data Register (SARADC_ICDR79)	32	R/W	0000_0000h	40.5.16/ 1639
240	Internal Channel Data Register (SARADC_ICDR80)	32	R/W	0000_0000h	40.5.16/ 1639

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
244	Internal Channel Data Register (SARADC_ICDR81)	32	R/W	0000_0000h	40.5.16/ 1639
248	Internal Channel Data Register (SARADC_ICDR82)	32	R/W	0000_0000h	40.5.16/ 1639
24C	Internal Channel Data Register (SARADC_ICDR83)	32	R/W	0000_0000h	40.5.16/ 1639
250	Internal Channel Data Register (SARADC_ICDR84)	32	R/W	0000_0000h	40.5.16/ 1639
254	Internal Channel Data Register (SARADC_ICDR85)	32	R/W	0000_0000h	40.5.16/ 1639
258	Internal Channel Data Register (SARADC_ICDR86)	32	R/W	0000_0000h	40.5.16/ 1639
25C	Internal Channel Data Register (SARADC_ICDR87)	32	R/W	0000_0000h	40.5.16/ 1639
260	Internal Channel Data Register (SARADC_ICDR88)	32	R/W	0000_0000h	40.5.16/ 1639
264	Internal Channel Data Register (SARADC_ICDR89)	32	R/W	0000_0000h	40.5.16/ 1639
268	Internal Channel Data Register (SARADC_ICDR90)	32	R/W	0000_0000h	40.5.16/ 1639
26C	Internal Channel Data Register (SARADC_ICDR91)	32	R/W	0000_0000h	40.5.16/ 1639
270	Internal Channel Data Register (SARADC_ICDR92)	32	R/W	0000_0000h	40.5.16/ 1639
274	Internal Channel Data Register (SARADC_ICDR93)	32	R/W	0000_0000h	40.5.16/ 1639
278	Internal Channel Data Register (SARADC_ICDR94)	32	R/W	0000_0000h	40.5.16/ 1639
27C	Internal Channel Data Register (SARADC_ICDR95)	32	R/W	0000_0000h	40.5.16/ 1639
2B0	Internal Channel Watchdog Selection Register (SARADC_ICWSELR0)	32	R/W	0000_0000h	40.5.17/ 1641
2B4	Internal Channel Watchdog Selection Register (SARADC_ICWSELR1)	32	R/W	0000_0000h	40.5.17/ 1641
2B8	Internal Channel Watchdog Selection Register (SARADC_ICWSELR2)	32	R/W	0000_0000h	40.5.17/ 1641
2BC	Internal Channel Watchdog Selection Register (SARADC_ICWSELR3)	32	R/W	0000_0000h	40.5.17/ 1641
2C0	Internal Channel Watchdog Selection Register (SARADC_ICWSELR4)	32	R/W	0000_0000h	40.5.17/ 1641
2C4	Internal Channel Watchdog Selection Register (SARADC_ICWSELR5)	32	R/W	0000_0000h	40.5.17/ 1641
2C8	Internal Channel Watchdog Selection Register (SARADC_ICWSELR6)	32	R/W	0000_0000h	40.5.17/ 1641

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2CC	Internal Channel Watchdog Selection Register (SARADC_ICWSELR7)	32	R/W	0000_0000h	40.5.17/ 1641
2D0	Internal Channel Watchdog Selection Register (SARADC_ICWSELR8)	32	R/W	0000_0000h	40.5.17/ 1641
2D4	Internal Channel Watchdog Selection Register (SARADC_ICWSELR9)	32	R/W	0000_0000h	40.5.17/ 1641
2D8	Internal Channel Watchdog Selection Register (SARADC_ICWSELR10)	32	R/W	0000_0000h	40.5.17/ 1641
2DC	Internal Channel Watchdog Selection Register (SARADC_ICWSELR11)	32	R/W	0000_0000h	40.5.17/ 1641
2E0	Internal Channel Watchdog Enable Register (SARADC_ICWENR0)	32	R/W	0000_0000h	40.5.18/ 1642
2E4	Internal Channel Watchdog Enable Register (SARADC_ICWENR1)	32	R/W	0000_0000h	40.5.18/ 1642
2E8	Internal Channel Watchdog Enable Register (SARADC_ICWENR2)	32	R/W	0000_0000h	40.5.18/ 1642
2F0	Internal Channel Analog Watchdog Out of Range register (SARADC_ICAWORR0)	32	w1c	0000_0000h	40.5.19/ 1643
2F4	Internal Channel Analog Watchdog Out of Range register (SARADC_ICAWORR1)	32	w1c	0000_0000h	40.5.19/ 1643
2F8	Internal Channel Analog Watchdog Out of Range register (SARADC_ICAWORR2)	32	w1c	0000_0000h	40.5.19/ 1643
400	Test Channel Interrupt Pending Register (SARADC_TCIPR)	32	w1c	0000_0000h	40.5.20/ 1644
404	Test Channel Interrupt Mask Register (SARADC_TCIMR)	32	R/W	0000_0000h	40.5.21/ 1644
408	Test Channel DMA Select Register (SARADC_TCDSR)	32	R/W	0000_0000h	40.5.22/ 1645
40C	Test Channel Normal Conversion Mask Register (SARADC_TCNCMR)	32	R/W	0000_0000h	40.5.23/ 1645
410	Test Channel Injected Conversion Mask Register (SARADC_TCJCMR)	32	R/W	0000_0000h	40.5.24/ 1646
414	Test Channel Watchdog Selection Register (SARADC_TCWSELR0)	32	R/W	0000_0000h	40.5.25/ 1646
418	Test Channel Watchdog Selection Register (SARADC_TCWSELR1)	32	R/W	0000_0000h	40.5.25/ 1646
41C	Test Channel Watchdog Selection Register (SARADC_TCWSELR2)	32	R/W	0000_0000h	40.5.25/ 1646
420	Test Channel Watchdog Selection Register (SARADC_TCWSELR3)	32	R/W	0000_0000h	40.5.25/ 1646
424	Test Channel Watchdog Enable Register (SARADC_TCWENR)	32	R/W	0000_0000h	40.5.26/ 1648
428	Test Channel Analog Watchdog Out of Range Register (SARADC_TCAWORR)	32	w1c	0000_0000h	40.5.27/ 1648

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
430	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR0)	32	R/W	0000_0000h	40.5.28/ 1649
434	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR1)	32	R/W	0000_0000h	40.5.28/ 1649
438	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR2)	32	R/W	0000_0000h	40.5.28/ 1649
43C	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR3)	32	R/W	0000_0000h	40.5.28/ 1649
440	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR4)	32	R/W	0000_0000h	40.5.28/ 1649
444	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR5)	32	R/W	0000_0000h	40.5.28/ 1649
448	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR6)	32	R/W	0000_0000h	40.5.28/ 1649
44C	Test Channel Connection with Analog Pin Register (SARADC_TCCAPR7)	32	R/W	0000_0000h	40.5.28/ 1649
450	Test Channel Data Register (SARADC_TCDR96)	32	R/W	0000_0000h	40.5.29/ 1651
454	Test Channel Data Register (SARADC_TCDR97)	32	R/W	0000_0000h	40.5.29/ 1651
458	Test Channel Data Register (SARADC_TCDR98)	32	R/W	0000_0000h	40.5.29/ 1651
45C	Test Channel Data Register (SARADC_TCDR99)	32	R/W	0000_0000h	40.5.29/ 1651
460	Test Channel Data Register (SARADC_TCDR100)	32	R/W	0000_0000h	40.5.29/ 1651
464	Test Channel Data Register (SARADC_TCDR101)	32	R/W	0000_0000h	40.5.29/ 1651
468	Test Channel Data Register (SARADC_TCDR102)	32	R/W	0000_0000h	40.5.29/ 1651
46C	Test Channel Data Register (SARADC_TCDR103)	32	R/W	0000_0000h	40.5.29/ 1651
470	Test Channel Data Register (SARADC_TCDR104)	32	R/W	0000_0000h	40.5.29/ 1651
474	Test Channel Data Register (SARADC_TCDR105)	32	R/W	0000_0000h	40.5.29/ 1651
478	Test Channel Data Register (SARADC_TCDR106)	32	R/W	0000_0000h	40.5.29/ 1651
47C	Test Channel Data Register (SARADC_TCDR107)	32	R/W	0000_0000h	40.5.29/ 1651
480	Test Channel Data Register (SARADC_TCDR108)	32	R/W	0000_0000h	40.5.29/ 1651
484	Test Channel Data Register (SARADC_TCDR109)	32	R/W	0000_0000h	40.5.29/ 1651

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
488	Test Channel Data Register (SARADC_TCDR110)	32	R/W	0000_0000h	40.5.29/ 1651
48C	Test Channel Data Register (SARADC_TCDR111)	32	R/W	0000_0000h	40.5.29/ 1651
490	Test Channel Data Register (SARADC_TCDR112)	32	R/W	0000_0000h	40.5.29/ 1651
494	Test Channel Data Register (SARADC_TCDR113)	32	R/W	0000_0000h	40.5.29/ 1651
498	Test Channel Data Register (SARADC_TCDR114)	32	R/W	0000_0000h	40.5.29/ 1651
49C	Test Channel Data Register (SARADC_TCDR115)	32	R/W	0000_0000h	40.5.29/ 1651
4A0	Test Channel Data Register (SARADC_TCDR116)	32	R/W	0000_0000h	40.5.29/ 1651
4A4	Test Channel Data Register (SARADC_TCDR117)	32	R/W	0000_0000h	40.5.29/ 1651
4A8	Test Channel Data Register (SARADC_TCDR118)	32	R/W	0000_0000h	40.5.29/ 1651
4AC	Test Channel Data Register (SARADC_TCDR119)	32	R/W	0000_0000h	40.5.29/ 1651
4B0	Test Channel Data Register (SARADC_TCDR120)	32	R/W	0000_0000h	40.5.29/ 1651
4B4	Test Channel Data Register (SARADC_TCDR121)	32	R/W	0000_0000h	40.5.29/ 1651
4B8	Test Channel Data Register (SARADC_TCDR122)	32	R/W	0000_0000h	40.5.29/ 1651
4BC	Test Channel Data Register (SARADC_TCDR123)	32	R/W	0000_0000h	40.5.29/ 1651
4C0	Test Channel Data Register (SARADC_TCDR124)	32	R/W	0000_0000h	40.5.29/ 1651
4C4	Test Channel Data Register (SARADC_TCDR125)	32	R/W	0000_0000h	40.5.29/ 1651
4C8	Test Channel Data Register (SARADC_TCDR126)	32	R/W	0000_0000h	40.5.29/ 1651
4CC	Test Channel Data Register (SARADC_TCDR127)	32	R/W	0000_0000h	40.5.29/ 1651
500	External Channel Decode Signals Delay Register (SARADC_ECDSDR)	32	R/W	0000_0000h	40.5.30/ 1653
510	External Channel Interrupt Pending Register (SARADC_ECIPR0)	32	w1c	0000_0000h	40.5.31/ 1653
514	External Channel Interrupt Pending Register (SARADC_ECIPR1)	32	w1c	0000_0000h	40.5.31/ 1653
518	External Channel Interrupt Pending Register (SARADC_ECIPR2)	32	w1c	0000_0000h	40.5.31/ 1653

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
51C	External Channel Interrupt Pending Register (SARADC_ECIPR3)	32	w1c	0000_0000h	40.5.31/ 1653
520	External Channel Interrupt Mask Register (SARADC_ECIMR0)	32	R/W	0000_0000h	40.5.32/ 1654
524	External Channel Interrupt Mask Register (SARADC_ECIMR1)	32	R/W	0000_0000h	40.5.32/ 1654
528	External Channel Interrupt Mask Register (SARADC_ECIMR2)	32	R/W	0000_0000h	40.5.32/ 1654
52C	External Channel Interrupt Mask Register (SARADC_ECIMR3)	32	R/W	0000_0000h	40.5.32/ 1654
530	External Channel DMA Select Register (SARADC_ECDSR0)	32	R/W	0000_0000h	40.5.33/ 1654
534	External Channel DMA Select Register (SARADC_ECDSR1)	32	R/W	0000_0000h	40.5.33/ 1654
538	External Channel DMA Select Register (SARADC_ECDSR2)	32	R/W	0000_0000h	40.5.33/ 1654
53C	External Channel DMA Select Register (SARADC_ECDSR3)	32	R/W	0000_0000h	40.5.33/ 1654
540	External Channel Normal Conversion Mask Register (SARADC_ECNCMR0)	32	R/W	0000_0000h	40.5.34/ 1655
544	External Channel Normal Conversion Mask Register (SARADC_ECNCMR1)	32	R/W	0000_0000h	40.5.34/ 1655
548	External Channel Normal Conversion Mask Register (SARADC_ECNCMR2)	32	R/W	0000_0000h	40.5.34/ 1655
54C	External Channel Normal Conversion Mask Register (SARADC_ECNCMR3)	32	R/W	0000_0000h	40.5.34/ 1655
550	External Channel Injected Conversion Mask Register (SARADC_ECJCMR0)	32	R/W	0000_0000h	40.5.35/ 1655
554	External Channel Injected Conversion Mask Register (SARADC_ECJCMR1)	32	R/W	0000_0000h	40.5.35/ 1655
558	External Channel Injected Conversion Mask Register (SARADC_ECJCMR2)	32	R/W	0000_0000h	40.5.35/ 1655
55C	External Channel Injected Conversion Mask Register (SARADC_ECJCMR3)	32	R/W	0000_0000h	40.5.35/ 1655
560	External Channel Watchdog Selection Register (SARADC_ECWSELR0)	32	R/W	0000_0000h	40.5.36/ 1656
564	External Channel Watchdog Selection Register (SARADC_ECWSELR1)	32	R/W	0000_0000h	40.5.36/ 1656
568	External Channel Watchdog Selection Register (SARADC_ECWSELR2)	32	R/W	0000_0000h	40.5.36/ 1656
56C	External Channel Watchdog Selection Register (SARADC_ECWSELR3)	32	R/W	0000_0000h	40.5.36/ 1656
570	External Channel Watchdog Selection Register (SARADC_ECWSELR4)	32	R/W	0000_0000h	40.5.36/ 1656

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
574	External Channel Watchdog Selection Register (SARADC_ECWSELR5)	32	R/W	0000_0000h	40.5.36/ 1656
578	External Channel Watchdog Selection Register (SARADC_ECWSELR6)	32	R/W	0000_0000h	40.5.36/ 1656
57C	External Channel Watchdog Selection Register (SARADC_ECWSELR7)	32	R/W	0000_0000h	40.5.36/ 1656
580	External Channel Watchdog Selection Register (SARADC_ECWSELR8)	32	R/W	0000_0000h	40.5.36/ 1656
584	External Channel Watchdog Selection Register (SARADC_ECWSELR9)	32	R/W	0000_0000h	40.5.36/ 1656
588	External Channel Watchdog Selection Register (SARADC_ECWSELR10)	32	R/W	0000_0000h	40.5.36/ 1656
58C	External Channel Watchdog Selection Register (SARADC_ECWSELR11)	32	R/W	0000_0000h	40.5.36/ 1656
590	External Channel Watchdog Selection Register (SARADC_ECWSELR12)	32	R/W	0000_0000h	40.5.36/ 1656
594	External Channel Watchdog Selection Register (SARADC_ECWSELR13)	32	R/W	0000_0000h	40.5.36/ 1656
598	External Channel Watchdog Selection Register (SARADC_ECWSELR14)	32	R/W	0000_0000h	40.5.36/ 1656
59C	External Channel Watchdog Selection Register (SARADC_ECWSELR15)	32	R/W	0000_0000h	40.5.36/ 1656
5A0	External Channel Watchdog Enable Register (SARADC_ECWENR0)	32	R/W	0000_0000h	40.5.37/ 1658
5A4	External Channel Watchdog Enable Register (SARADC_ECWENR1)	32	R/W	0000_0000h	40.5.37/ 1658
5A8	External Channel Watchdog Enable Register (SARADC_ECWENR2)	32	R/W	0000_0000h	40.5.37/ 1658
5AC	External Channel Watchdog Enable Register (SARADC_ECWENR3)	32	R/W	0000_0000h	40.5.37/ 1658
5B0	External Channel Analog Watchdog Out of Range register (SARADC_ECAWORR0)	32	w1c	0000_0000h	40.5.38/ 1658
5B4	External Channel Analog Watchdog Out of Range register (SARADC_ECAWORR1)	32	w1c	0000_0000h	40.5.38/ 1658
5B8	External Channel Analog Watchdog Out of Range register (SARADC_ECAWORR2)	32	w1c	0000_0000h	40.5.38/ 1658
5BC	External Channel Analog Watchdog Out of Range register (SARADC_ECAWORR3)	32	w1c	0000_0000h	40.5.38/ 1658
5C0	External Channel Mapping to Internal Channel Register (SARADC_ECMICR0)	32	R/W	0000_0000h	40.5.39/ 1659
5C4	External Channel Mapping to Internal Channel Register (SARADC_ECMICR1)	32	R/W	0000_0000h	40.5.39/ 1659
5C8	External Channel Mapping to Internal Channel Register (SARADC_ECMICR2)	32	R/W	0000_0000h	40.5.39/ 1659

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5CC	External Channel Mapping to Internal Channel Register (SARADC_ECMICR3)	32	R/W	0000_0000h	40.5.39/ 1659
5D0	External Channel Data Register (SARADC_ECDR128)	32	R/W	0000_0000h	40.5.40/ 1660
5D4	External Channel Data Register (SARADC_ECDR129)	32	R/W	0000_0000h	40.5.40/ 1660
5D8	External Channel Data Register (SARADC_ECDR130)	32	R/W	0000_0000h	40.5.40/ 1660
5DC	External Channel Data Register (SARADC_ECDR131)	32	R/W	0000_0000h	40.5.40/ 1660
5E0	External Channel Data Register (SARADC_ECDR132)	32	R/W	0000_0000h	40.5.40/ 1660
5E4	External Channel Data Register (SARADC_ECDR133)	32	R/W	0000_0000h	40.5.40/ 1660
5E8	External Channel Data Register (SARADC_ECDR134)	32	R/W	0000_0000h	40.5.40/ 1660
5EC	External Channel Data Register (SARADC_ECDR135)	32	R/W	0000_0000h	40.5.40/ 1660
5F0	External Channel Data Register (SARADC_ECDR136)	32	R/W	0000_0000h	40.5.40/ 1660
5F4	External Channel Data Register (SARADC_ECDR137)	32	R/W	0000_0000h	40.5.40/ 1660
5F8	External Channel Data Register (SARADC_ECDR138)	32	R/W	0000_0000h	40.5.40/ 1660
5FC	External Channel Data Register (SARADC_ECDR139)	32	R/W	0000_0000h	40.5.40/ 1660
600	External Channel Data Register (SARADC_ECDR140)	32	R/W	0000_0000h	40.5.40/ 1660
604	External Channel Data Register (SARADC_ECDR141)	32	R/W	0000_0000h	40.5.40/ 1660
608	External Channel Data Register (SARADC_ECDR142)	32	R/W	0000_0000h	40.5.40/ 1660
60C	External Channel Data Register (SARADC_ECDR143)	32	R/W	0000_0000h	40.5.40/ 1660
610	External Channel Data Register (SARADC_ECDR144)	32	R/W	0000_0000h	40.5.40/ 1660
614	External Channel Data Register (SARADC_ECDR145)	32	R/W	0000_0000h	40.5.40/ 1660
618	External Channel Data Register (SARADC_ECDR146)	32	R/W	0000_0000h	40.5.40/ 1660
61C	External Channel Data Register (SARADC_ECDR147)	32	R/W	0000_0000h	40.5.40/ 1660
620	External Channel Data Register (SARADC_ECDR148)	32	R/W	0000_0000h	40.5.40/ 1660

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
624	External Channel Data Register (SARADC_ECDR149)	32	R/W	0000_0000h	40.5.40/ 1660
628	External Channel Data Register (SARADC_ECDR150)	32	R/W	0000_0000h	40.5.40/ 1660
62C	External Channel Data Register (SARADC_ECDR151)	32	R/W	0000_0000h	40.5.40/ 1660
630	External Channel Data Register (SARADC_ECDR152)	32	R/W	0000_0000h	40.5.40/ 1660
634	External Channel Data Register (SARADC_ECDR153)	32	R/W	0000_0000h	40.5.40/ 1660
638	External Channel Data Register (SARADC_ECDR154)	32	R/W	0000_0000h	40.5.40/ 1660
63C	External Channel Data Register (SARADC_ECDR155)	32	R/W	0000_0000h	40.5.40/ 1660
640	External Channel Data Register (SARADC_ECDR156)	32	R/W	0000_0000h	40.5.40/ 1660
644	External Channel Data Register (SARADC_ECDR157)	32	R/W	0000_0000h	40.5.40/ 1660
648	External Channel Data Register (SARADC_ECDR158)	32	R/W	0000_0000h	40.5.40/ 1660
64C	External Channel Data Register (SARADC_ECDR159)	32	R/W	0000_0000h	40.5.40/ 1660
650	External Channel Data Register (SARADC_ECDR160)	32	R/W	0000_0000h	40.5.40/ 1660
654	External Channel Data Register (SARADC_ECDR161)	32	R/W	0000_0000h	40.5.40/ 1660
658	External Channel Data Register (SARADC_ECDR162)	32	R/W	0000_0000h	40.5.40/ 1660
65C	External Channel Data Register (SARADC_ECDR163)	32	R/W	0000_0000h	40.5.40/ 1660
660	External Channel Data Register (SARADC_ECDR164)	32	R/W	0000_0000h	40.5.40/ 1660
664	External Channel Data Register (SARADC_ECDR165)	32	R/W	0000_0000h	40.5.40/ 1660
668	External Channel Data Register (SARADC_ECDR166)	32	R/W	0000_0000h	40.5.40/ 1660
66C	External Channel Data Register (SARADC_ECDR167)	32	R/W	0000_0000h	40.5.40/ 1660
670	External Channel Data Register (SARADC_ECDR168)	32	R/W	0000_0000h	40.5.40/ 1660
674	External Channel Data Register (SARADC_ECDR169)	32	R/W	0000_0000h	40.5.40/ 1660
678	External Channel Data Register (SARADC_ECDR170)	32	R/W	0000_0000h	40.5.40/ 1660

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
67C	External Channel Data Register (SARADC_ECDR171)	32	R/W	0000_0000h	40.5.40/ 1660
680	External Channel Data Register (SARADC_ECDR172)	32	R/W	0000_0000h	40.5.40/ 1660
684	External Channel Data Register (SARADC_ECDR173)	32	R/W	0000_0000h	40.5.40/ 1660
688	External Channel Data Register (SARADC_ECDR174)	32	R/W	0000_0000h	40.5.40/ 1660
68C	External Channel Data Register (SARADC_ECDR175)	32	R/W	0000_0000h	40.5.40/ 1660
690	External Channel Data Register (SARADC_ECDR176)	32	R/W	0000_0000h	40.5.40/ 1660
694	External Channel Data Register (SARADC_ECDR177)	32	R/W	0000_0000h	40.5.40/ 1660
698	External Channel Data Register (SARADC_ECDR178)	32	R/W	0000_0000h	40.5.40/ 1660
69C	External Channel Data Register (SARADC_ECDR179)	32	R/W	0000_0000h	40.5.40/ 1660
6A0	External Channel Data Register (SARADC_ECDR180)	32	R/W	0000_0000h	40.5.40/ 1660
6A4	External Channel Data Register (SARADC_ECDR181)	32	R/W	0000_0000h	40.5.40/ 1660
6A8	External Channel Data Register (SARADC_ECDR182)	32	R/W	0000_0000h	40.5.40/ 1660
6AC	External Channel Data Register (SARADC_ECDR183)	32	R/W	0000_0000h	40.5.40/ 1660
6B0	External Channel Data Register (SARADC_ECDR184)	32	R/W	0000_0000h	40.5.40/ 1660
6B4	External Channel Data Register (SARADC_ECDR185)	32	R/W	0000_0000h	40.5.40/ 1660
6B8	External Channel Data Register (SARADC_ECDR186)	32	R/W	0000_0000h	40.5.40/ 1660
6BC	External Channel Data Register (SARADC_ECDR187)	32	R/W	0000_0000h	40.5.40/ 1660
6C0	External Channel Data Register (SARADC_ECDR188)	32	R/W	0000_0000h	40.5.40/ 1660
6C4	External Channel Data Register (SARADC_ECDR189)	32	R/W	0000_0000h	40.5.40/ 1660
6C8	External Channel Data Register (SARADC_ECDR190)	32	R/W	0000_0000h	40.5.40/ 1660
6CC	External Channel Data Register (SARADC_ECDR191)	32	R/W	0000_0000h	40.5.40/ 1660
6D0	External Channel Data Register (SARADC_ECDR192)	32	R/W	0000_0000h	40.5.40/ 1660

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6D4	External Channel Data Register (SARADC_ECDR193)	32	R/W	0000_0000h	40.5.40/ 1660
6D8	External Channel Data Register (SARADC_ECDR194)	32	R/W	0000_0000h	40.5.40/ 1660
6DC	External Channel Data Register (SARADC_ECDR195)	32	R/W	0000_0000h	40.5.40/ 1660
6E0	External Channel Data Register (SARADC_ECDR196)	32	R/W	0000_0000h	40.5.40/ 1660
6E4	External Channel Data Register (SARADC_ECDR197)	32	R/W	0000_0000h	40.5.40/ 1660
6E8	External Channel Data Register (SARADC_ECDR198)	32	R/W	0000_0000h	40.5.40/ 1660
6EC	External Channel Data Register (SARADC_ECDR199)	32	R/W	0000_0000h	40.5.40/ 1660
6F0	External Channel Data Register (SARADC_ECDR200)	32	R/W	0000_0000h	40.5.40/ 1660
6F4	External Channel Data Register (SARADC_ECDR201)	32	R/W	0000_0000h	40.5.40/ 1660
6F8	External Channel Data Register (SARADC_ECDR202)	32	R/W	0000_0000h	40.5.40/ 1660
6FC	External Channel Data Register (SARADC_ECDR203)	32	R/W	0000_0000h	40.5.40/ 1660
700	External Channel Data Register (SARADC_ECDR204)	32	R/W	0000_0000h	40.5.40/ 1660
704	External Channel Data Register (SARADC_ECDR205)	32	R/W	0000_0000h	40.5.40/ 1660
708	External Channel Data Register (SARADC_ECDR206)	32	R/W	0000_0000h	40.5.40/ 1660
70C	External Channel Data Register (SARADC_ECDR207)	32	R/W	0000_0000h	40.5.40/ 1660
710	External Channel Data Register (SARADC_ECDR208)	32	R/W	0000_0000h	40.5.40/ 1660
714	External Channel Data Register (SARADC_ECDR209)	32	R/W	0000_0000h	40.5.40/ 1660
718	External Channel Data Register (SARADC_ECDR210)	32	R/W	0000_0000h	40.5.40/ 1660
71C	External Channel Data Register (SARADC_ECDR211)	32	R/W	0000_0000h	40.5.40/ 1660
720	External Channel Data Register (SARADC_ECDR212)	32	R/W	0000_0000h	40.5.40/ 1660
724	External Channel Data Register (SARADC_ECDR213)	32	R/W	0000_0000h	40.5.40/ 1660
728	External Channel Data Register (SARADC_ECDR214)	32	R/W	0000_0000h	40.5.40/ 1660

Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
72C	External Channel Data Register (SARADC_ECDR215)	32	R/W	0000_0000h	40.5.40/ 1660
730	External Channel Data Register (SARADC_ECDR216)	32	R/W	0000_0000h	40.5.40/ 1660
734	External Channel Data Register (SARADC_ECDR217)	32	R/W	0000_0000h	40.5.40/ 1660
738	External Channel Data Register (SARADC_ECDR218)	32	R/W	0000_0000h	40.5.40/ 1660
73C	External Channel Data Register (SARADC_ECDR219)	32	R/W	0000_0000h	40.5.40/ 1660
740	External Channel Data Register (SARADC_ECDR220)	32	R/W	0000_0000h	40.5.40/ 1660
744	External Channel Data Register (SARADC_ECDR221)	32	R/W	0000_0000h	40.5.40/ 1660
748	External Channel Data Register (SARADC_ECDR222)	32	R/W	0000_0000h	40.5.40/ 1660
74C	External Channel Data Register (SARADC_ECDR223)	32	R/W	0000_0000h	40.5.40/ 1660
750	External Channel Data Register (SARADC_ECDR224)	32	R/W	0000_0000h	40.5.40/ 1660
754	External Channel Data Register (SARADC_ECDR225)	32	R/W	0000_0000h	40.5.40/ 1660
758	External Channel Data Register (SARADC_ECDR226)	32	R/W	0000_0000h	40.5.40/ 1660
75C	External Channel Data Register (SARADC_ECDR227)	32	R/W	0000_0000h	40.5.40/ 1660
760	External Channel Data Register (SARADC_ECDR228)	32	R/W	0000_0000h	40.5.40/ 1660
764	External Channel Data Register (SARADC_ECDR229)	32	R/W	0000_0000h	40.5.40/ 1660
768	External Channel Data Register (SARADC_ECDR230)	32	R/W	0000_0000h	40.5.40/ 1660
76C	External Channel Data Register (SARADC_ECDR231)	32	R/W	0000_0000h	40.5.40/ 1660
770	External Channel Data Register (SARADC_ECDR232)	32	R/W	0000_0000h	40.5.40/ 1660
774	External Channel Data Register (SARADC_ECDR233)	32	R/W	0000_0000h	40.5.40/ 1660
778	External Channel Data Register (SARADC_ECDR234)	32	R/W	0000_0000h	40.5.40/ 1660
77C	External Channel Data Register (SARADC_ECDR235)	32	R/W	0000_0000h	40.5.40/ 1660
780	External Channel Data Register (SARADC_ECDR236)	32	R/W	0000_0000h	40.5.40/ 1660

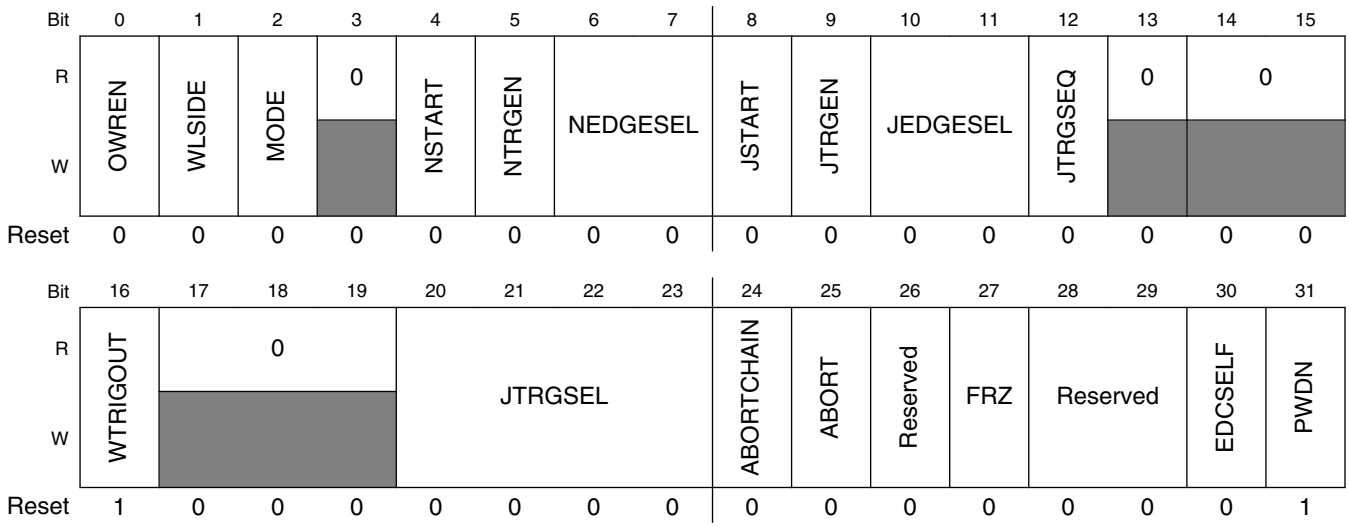
Table continues on the next page...

SARADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
784	External Channel Data Register (SARADC_ECDR237)	32	R/W	0000_0000h	40.5.40/ 1660
788	External Channel Data Register (SARADC_ECDR238)	32	R/W	0000_0000h	40.5.40/ 1660
78C	External Channel Data Register (SARADC_ECDR239)	32	R/W	0000_0000h	40.5.40/ 1660
790	External Channel Data Register (SARADC_ECDR240)	32	R/W	0000_0000h	40.5.40/ 1660
794	External Channel Data Register (SARADC_ECDR241)	32	R/W	0000_0000h	40.5.40/ 1660
798	External Channel Data Register (SARADC_ECDR242)	32	R/W	0000_0000h	40.5.40/ 1660
79C	External Channel Data Register (SARADC_ECDR243)	32	R/W	0000_0000h	40.5.40/ 1660
7A0	External Channel Data Register (SARADC_ECDR244)	32	R/W	0000_0000h	40.5.40/ 1660
7A4	External Channel Data Register (SARADC_ECDR245)	32	R/W	0000_0000h	40.5.40/ 1660
7A8	External Channel Data Register (SARADC_ECDR246)	32	R/W	0000_0000h	40.5.40/ 1660
7AC	External Channel Data Register (SARADC_ECDR247)	32	R/W	0000_0000h	40.5.40/ 1660
7B0	External Channel Data Register (SARADC_ECDR248)	32	R/W	0000_0000h	40.5.40/ 1660
7B4	External Channel Data Register (SARADC_ECDR249)	32	R/W	0000_0000h	40.5.40/ 1660
7B8	External Channel Data Register (SARADC_ECDR250)	32	R/W	0000_0000h	40.5.40/ 1660
7BC	External Channel Data Register (SARADC_ECDR251)	32	R/W	0000_0000h	40.5.40/ 1660
7C0	External Channel Data Register (SARADC_ECDR252)	32	R/W	0000_0000h	40.5.40/ 1660
7C4	External Channel Data Register (SARADC_ECDR253)	32	R/W	0000_0000h	40.5.40/ 1660
7C8	External Channel Data Register (SARADC_ECDR254)	32	R/W	0000_0000h	40.5.40/ 1660
7CC	External Channel Data Register (SARADC_ECDR255)	32	R/W	0000_0000h	40.5.40/ 1660

40.5.1 Main Configuration Register (SARADC_MCR)

Address: 1000h base + 0h offset = 1000h



SARADC_MCR field descriptions

Field	Description
0 OWREN	Overwrite enable This bit enables or disables the functionality to overwrite unread converted data in ICDR, TCDR, ECDR registers. 0 Prevents overwrite of unread converted data, new result is discarded 1 Enables converted data to be overwritten by a new conversion
1 WLSIDE	Write left/right-aligned 0 The conversion data in ICDR, TCDR, ECDR registers is written right-aligned. 1 Conversion data is left-aligned (from 15 to (15 - resolution + 1)).
2 MODE	One Shot/Scan 0 One Shot Mode-Configures the normal conversion of one chain. 1 Scan Mode-Configures continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 NSTART	Normal Start conversion Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation. 0 Causes the current chain conversion to finish and stops the operation 1 Starts the chain or scan conversion
5 NTRGEN	Normal trigger enable

Table continues on the next page...

SARADC_MCR field descriptions (continued)

Field	Description
	0 Normal trigger disabled to start a normal conversion. 1 Normal trigger enabled to start a normal conversion.
6–7 NEDGESEL	Normal trigger edge selection 00 Falling edge of normal trigger is selected. 01 Rising edge of normal trigger is selected. 10 Both rising and falling edges of normal trigger are selected. 11 Same as 10b.
8 JSTART	Injected Start conversion Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted. 0 Writing '0' has no effect. 1 Starts the injected chain conversion
9 JTRGEN	Injection trigger enable 0 Injection trigger disabled for channel injection (injected conversion cannot be started using an injection trigger) 1 Injection trigger enabled for channel injection
10–11 JEDGESEL	Injection trigger edge selection 00 Falling edge of injection trigger is selected. 01 Rising edge of injection trigger is selected. 10 Both rising and falling edges of injection trigger are selected. 11 Same as 10b.
12 JTRGSEQ	Injection trigger sequence enable 0 Injection trigger sequence mode is disabled. 1 Injection trigger sequence mode is enabled.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 WTRIGOUT	WDG Crossover External Trigger Output This control bit enables the Trigger on every threshold crossover event(crossing of Lower/Upper thresholds) of each WDG monitor to external trigger port(up to 16 ports for up to 16 WDGs). 0 Trigger Outputs enabled 1 Trigger Outputs disabled
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 JTRGSEL	Injection Trigger Input Selection This bitfield selects which trigger input to be selected to start injected conversion. 0000 Trigger input 0 is selected 0001 Trigger input 1 is selected 0010 Trigger input 2 is selected 0011 Trigger input 3 is selected

Table continues on the next page...

SARADC_MCR field descriptions (continued)

Field	Description
	0100 Trigger input 4 is selected 0101 Trigger input 5 is selected 0110 Trigger input 6 is selected 0111 Trigger input 7 is selected 1000 Trigger input 8 is selected 1001 Trigger input 9 is selected 1010 Trigger input 10 is selected 1011 Trigger input 11 is selected 1100 Trigger input 12 is selected 1101 Trigger input 13 is selected 1110 Trigger input 14 is selected 1111 Trigger input 15 is selected.
24 ABORTCHAIN	Abort Chain When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested. 0 Conversion is not affected 1 Aborts the ongoing chain conversion
25 ABORT	Abort Conversion When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked. 0 Conversion is not affected 1 Aborts the ongoing conversion
26 Reserved	This field is reserved. Reserved
27 FRZ	Freeze This bit enables to stop the SARADC conversions at the end of current channel conversion when SoC enters debug mode. 0 Conversions are not stopped. 1 When the chip enters debug mode, further conversions by SARADC analog block will be stopped. The ongoing channel conversion will be completed and the converted data gets stored. When the SoC exits debug mode, SARADC resumes the job that was left before halting the conversions. Changing configuration registers during halted condition may cause a pending operation to fail when normal operation is resumed.
28–29 Reserved	This field is reserved. Reserved
30 EDCSELF	External Decode Channel Format Select This bit selects the format for 3-bit output port ipp_decode_extch used as select line for external decode channel 8:1 muxes. The last three bits of external channel number are passed as is (if value 0) or converted to gray code (if value 1) and passed to this port. 0 Binary format. 1 Gray Code format.
31 PWDN	Power-down enable

Table continues on the next page...

SARADC_MCR field descriptions (continued)

Field	Description
	When this bit is set, the analog module is requested to enter Power Down mode. When SARADC status is PWDN, resetting this bit starts SARADC transition to IDLE mode. 0 SARADC is in normal mode 1 SARADC has been requested to power down

40.5.2 Main Status Register (SARADC_MSR)

Address: 1000h base + 4h offset = 1004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				NSTART	Reserved			JSTART	Reserved				JABORTCHAIN	Reserved	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR								Reserved					ADCSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SARADC_MSR field descriptions

Field	Description
0–3 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
4 NSTART	This status bit is used to signal that a normal conversion is ongoing. 0 Normal conversion is not taking place. 1 Normal conversion is ongoing or the normal conversion is pending due to injected conversion or CTU triggered injected conversion.
5–7 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
8 JSTART	This status bit is used to signal that a injected conversion is ongoing. 0 Injected conversion is not taking place. 1 Injected conversion is ongoing.
9–12 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
13 JABORTCHAIN	This status bit is used to signal that an Injected conversion chain has been aborted. This bit is reset when a new conversion starts. 0 Last injected conversion chain has not been aborted. 1 Last injected conversion chain has been aborted.
14 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 CHADDR	Channel under measure address (CHADDR[7:0]) This status bit is used to signal which channel is under measure.
24–28 Reserved	This field is reserved. Reserved
29–31 ADCSTATUS	ADCSTATUS[2:0]. The value of this field depends on ADC status. While requesting an external channel conversion, the SARADC digital interface needs to wait for a fixed time determined by DSD bitfield of DSDR before proceeding to actual sampling phase. This is known as wait state. 000 IDLE 001 Power-down 010 Wait state 011 - 100 Sample 101 - 110 Conversion 111 -

40.5.3 Interrupt Status Register (SARADC_ISR)

Address: 1000h base + 10h offset = 1010h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											0	JEOC	JECH	NEOC	NECH
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ISR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 JEOC	End of injected channel conversion interrupt flag This bit indicates the end of conversion for an injected channel. 0 End of conversion of injected channel has not occurred since the last time the flag is cleared. 1 End of conversion of injected channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
29 JECH	End of injected chain conversion interrupt flag This bit indicates the end of conversion for an injected chain. 0 End of conversion of injected chain has not occurred since the last time the flag is cleared. 1 End of conversion of injected chain has occurred or the flag has not been cleared since the last end of conversion occurrence.
30 NEOC	End of normal channel conversion interrupt flag This bit indicates the end of conversion for a normal channel. 0 End of conversion of normal channel has not occurred since the last time the flag is cleared. 1 End of conversion of normal channel has occurred or the flag has not been cleared since the last end of conversion occurrence.

Table continues on the next page...

SARADC_ISR field descriptions (continued)

Field	Description
31 NECH	End of normal chain conversion interrupt flag This bit indicates the end of conversion for an normal chain. 0 End of conversion of normal chain has not occurred since the last time the flag is cleared. 1 End of conversion of normal chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

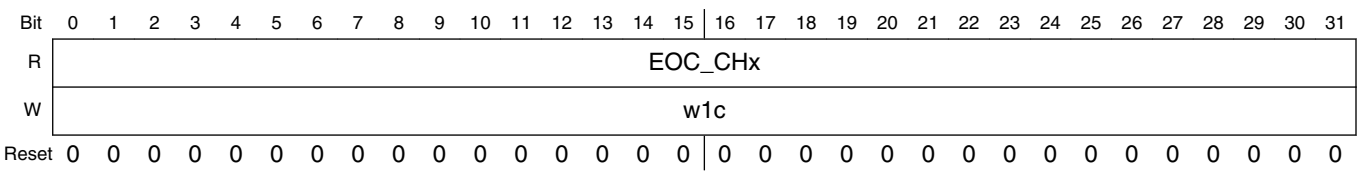
40.5.4 Internal channel Interrupt Pending Register (SARADC_ICIPR*n*)

The interrupt channel register to channel association is described below.

Table 40-2. ICIPR0-2 Registers to channel association

Register	Register Bits 0:31
ICIPR0	EOC_CH[31:0]
ICIPR1	EOC_CH[63:32]
ICIPR2	EOC_CH[95:64]

Address: 1000h base + 14h offset + (4d × i), where i=0d to 2d



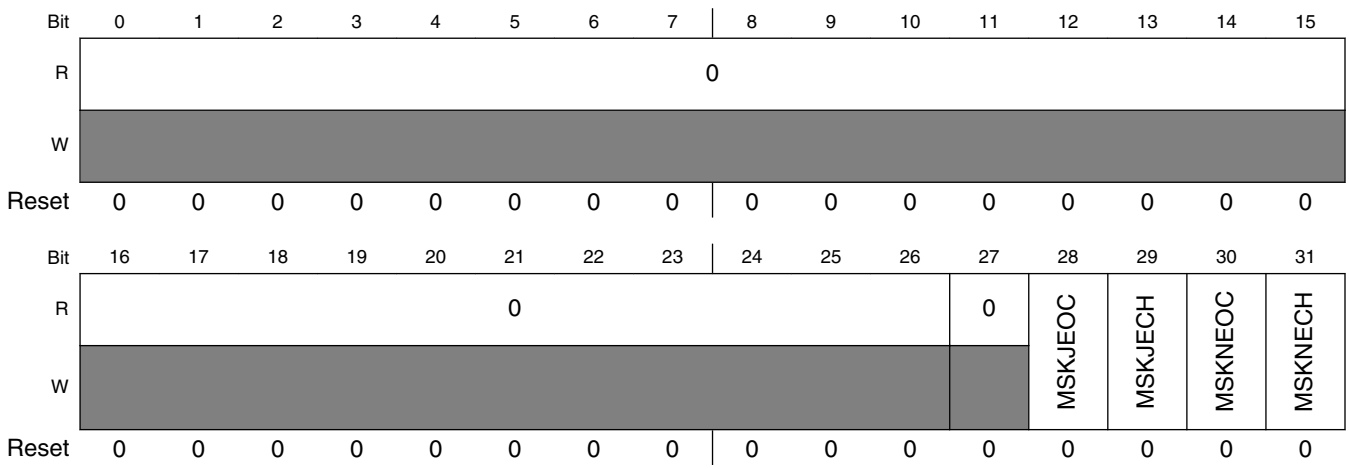
SARADC_ICIPR*n* field descriptions

Field	Description
0–31 EOC_CHx	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

40.5.5 Interrupt Mask Register (SARADC_IMR)

The description of the Interrupt Mask Register is given below.

Address: 1000h base + 20h offset = 1020h



SARADC_IMR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 MSKJEOC	Mask bit for JEOC 0 JEOC interrupt is disabled. 1 JEOC interrupt is enabled.
29 MSKJECH	Mask bit for JECH 0 JECH interrupt is disabled. 1 JECH interrupt is enabled.
30 MSKNEOC	Mask bit for NEOC 0 NEOC interrupt is disabled. 1 NEOC interrupt is enabled.
31 MSKNECH	Mask bit for NECH 0 NECH interrupt is disabled. 1 NECH interrupt is enabled.

40.5.6 Internal Channel Interrupt Mask Register (SARADC_ICIMR*n*)

The interrupt mask register to channel association is described below.

Table 40-3. Interrupt Mask Registers to Channel Association

Register	Register Bits 0:31
ICIMR0	IM_CH[31:0]
ICIMR1	IM_CH[63:32]
ICIMR2	IM_CH[95:64]

Address: 1000h base + 24h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ICIMR*n* field descriptions

Field	Description
0–31 IM_CHx	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

40.5.7 Watchdog Threshold Interrupt Status Register (SARADC_WTISR)

This register gives the interrupt status information for the 16 possible upper/lower threshold limits which can be selected for each channel.

NOTE

This register is implemented only on ADCs that have analog watchdogs.

Address: 1000h base + 30h offset = 1030h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_WTISR field descriptions

Field	Description
0 WDG15H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
1 WDG15L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
2 WDG14H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
3 WDG14L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
4 WDG13H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.

Table continues on the next page...

SARADC_WTISR field descriptions (continued)

Field	Description
5 WDG13L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
6 WDG12H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
7 WDG12L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
8 WDG11H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
9 WDG11L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
10 WDG10H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
11 WDG10L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
12 WDG9H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
13 WDG9L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
14 WDG8H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.

Table continues on the next page...

SARADC_WTISR field descriptions (continued)

Field	Description
15 WDG8L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
16 WDG7H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
17 WDG7L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
18 WDG6H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
19 WDG6L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
20 WDG5H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
21 WDG5L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
22 WDG4H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
23 WDG4L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
24 WDG3H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.

Table continues on the next page...

SARADC_WTISR field descriptions (continued)

Field	Description
25 WDG3L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
26 WDG2H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
27 WDG2L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
28 WDG1H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
29 WDG1L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.
30 WDG0H	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
31 WDG0L	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.

40.5.8 Watchdog Threshold Interrupt Mask Register (SARADC_WTIMR)

This register gives the interrupt mask information for the 16 possible upper/lower threshold limits which can be selected for each channel.

NOTE

This register is not implemented only on ADCs that have analog watchdogs.

Address: 1000h base + 34h offset = 1034h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_WTIMR field descriptions

Field	Description
0 MSKWDG15H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG15H is disabled. 1 Interrupt for WDG15H is enabled.
1 MSKWDG15L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG15L is disabled. 1 Interrupt for WDG15L is enabled.
2 MSKWDG14H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG14H is disabled. 1 Interrupt for WDG14H is enabled.
3 MSKWDG14L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG14L is disabled. 1 Interrupt for WDG14L is enabled.
4 MSKWDG13H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG13H is disabled. 1 Interrupt for WDG13H is enabled.
5 MSKWDG13L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG13L is disabled. 1 Interrupt for WDG13L is enabled.
6 MSKWDG12H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold.

Table continues on the next page...

SARADC_WTIMR field descriptions (continued)

Field	Description
	0 Interrupt for WDG12H is disabled. 1 Interrupt for WDG12H is enabled.
7 MSKWDG12L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG12L is disabled. 1 Interrupt for WDG12L is enabled.
8 MSKWDG11H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG11H is disabled. 1 Interrupt for WDG11H is enabled.
9 MSKWDG11L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG11L is disabled. 1 Interrupt for WDG11L is enabled.
10 MSKWDG10H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG10H is disabled. 1 Interrupt for WDG10H is enabled.
11 MSKWDG10L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG10L is disabled. 1 Interrupt for WDG10L is enabled.
12 MSKWDG9H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG9H is disabled. 1 Interrupt for WDG9H is enabled.
13 MSKWDG9L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG9L is disabled. 1 Interrupt for WDG9L is enabled.
14 MSKWDG8H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG8H is disabled. 1 Interrupt for WDG8H is enabled.
15 MSKWDG8L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG8L is disabled. 1 Interrupt for WDG8L is enabled.
16 MSKWDG7H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold.

Table continues on the next page...

SARADC_WTIMR field descriptions (continued)

Field	Description
	0 Interrupt for WDG7H is disabled. 1 Interrupt for WDG7H is enabled.
17 MSKWDG7L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG7L is disabled. 1 Interrupt for WDG7L is enabled.
18 MSKWDG6H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG6H is disabled. 1 Interrupt for WDG6H is enabled.
19 MSKWDG6L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG6L is disabled. 1 Interrupt for WDG6L is enabled.
20 MSKWDG5H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG5H is disabled. 1 Interrupt for WDG5H is enabled.
21 MSKWDG5L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG5L is disabled. 1 Interrupt for WDG5L is enabled.
22 MSKWDG4H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG4H is disabled. 1 Interrupt for WDG4H is enabled.
23 MSKWDG4L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG4L is disabled. 1 Interrupt for WDG4L is enabled.
24 MSKWDG3H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG3H is disabled. 1 Interrupt for WDG3H is enabled.
25 MSKWDG3L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG3L is disabled. 1 Interrupt for WDG3L is enabled.
26 MSKWDG2H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold.

Table continues on the next page...

SARADC_WTIMR field descriptions (continued)

Field	Description
	0 Interrupt for WDG2H is disabled. 1 Interrupt for WDG2H is enabled.
27 MSKWDG2L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG2L is disabled. 1 Interrupt for WDG2L is enabled.
28 MSKWDG1H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG1H is disabled. 1 Interrupt for WDG1H is enabled.
29 MSKWDG1L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG1L is disabled. 1 Interrupt for WDG1L is enabled.
30 MSKWDG0H	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDG0H is disabled. 1 Interrupt for WDG0H is enabled.
31 MSKWDG0L	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDG0L is disabled. 1 Interrupt for WDG0L is enabled.

40.5.9 DMA Enable Register (SARADC_DMAE)

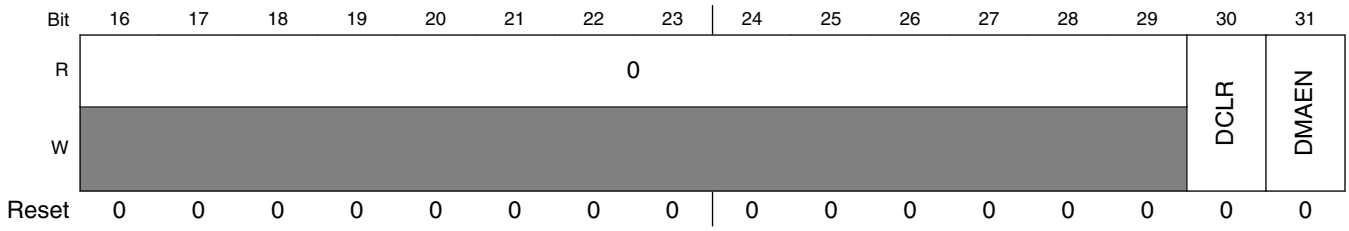
The DMA Enable (DMAE) register sets up the DMA for use with the SARADC.

NOTE

This register is implemented only for ADCs with DMA.

Address: 1000h base + 40h offset = 1040h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SARADC_DMAE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DCLR	DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers
31 DMAEN	DMA global enable 0 DMA feature disabled 1 DMA feature enabled

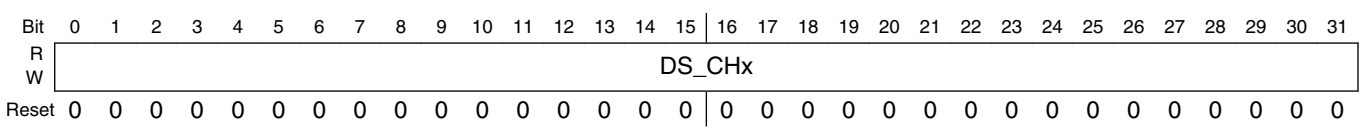
40.5.10 Internal Channel DMA Select Register (SARADC_ICDSR*n*)

The DMA select registers to channel association is described below.

Table 40-4. Internal Channel DMA Select Registers to Channel Association

Register	Register Bits 0:31
ICDSR0	DS_CH[31:0]
ICDSR1	DS_CH[63:32]
ICDSR2	DS_CH[95:64]

Address: 1000h base + 44h offset + (4d × i), where i=0d to 2d



SARADC_ICDSR*n* field descriptions

Field	Description
0–31 DS_CHx	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

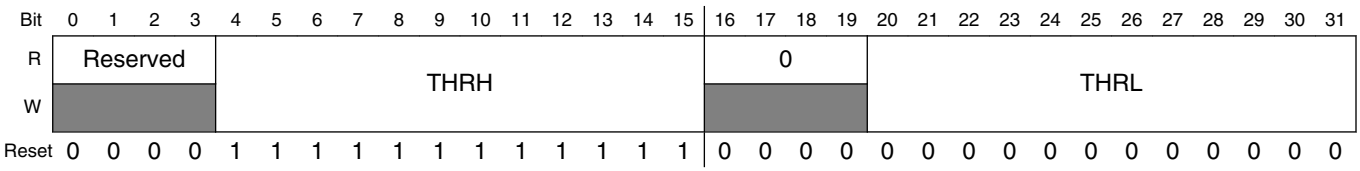
40.5.11 Watchdog Threshold Register (SARADC_WTHRHLR*n*)

NOTE

The number of WTHRHLR registers corresponds to the number of analog watchdogs implemented in the ADC module.

The length of the threshold field depends on the generic resolution.

Address: 1000h base + 60h offset + (4d × i), where i=0d to 3d

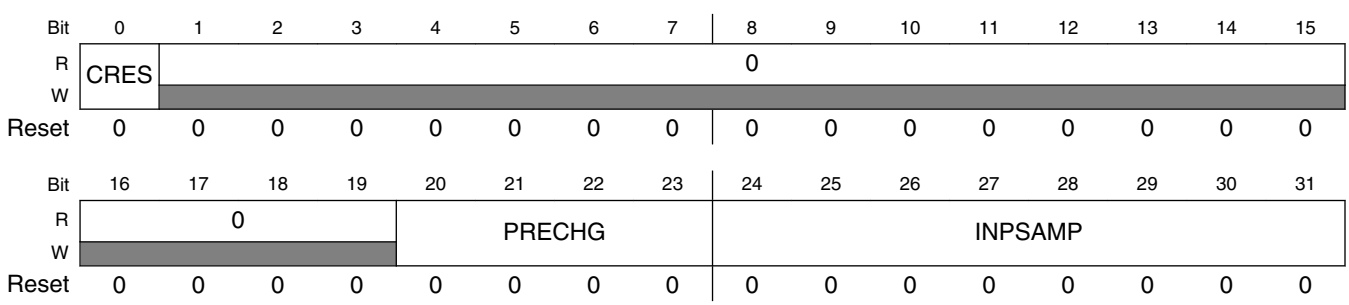


SARADC_WTHRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
4–15 THRH	High threshold value for channel x
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x

40.5.12 Conversion Timing Register (SARADC_CTR*n*)

Address: 1000h base + 94h offset + (4d × i), where i=0d to 3d



SARADC_CTR*n* field descriptions

Field	Description
0 CRES	Conversion Resolution select This bit selects the conversion resolution for the conversion.

Table continues on the next page...

SARADC_CTR*n* field descriptions (continued)

Field	Description
	0 12-bit resolution 1 10-bit resolution
1–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 PRECHG	Precharging phase duration This bitfield defines the precharging phase duration to be applied when this phase is enabled by PCE bit of channel data register. This duration is calculated as (PRECHG X 1/Frequency of SARADC clock). Please refer to device datasheet for the minimum duration required for precharging phase.
24–31 INPSAMP	Sampling phase duration This bitfield defines the sampling phase duration to be applied. This duration is calculated as (INPSAMP X 1/Frequency of SARADC clock). Please refer to device datasheet for the minimum duration required for sampling phase.

40.5.13 Internal Channel Normal Conversion Mask Register (SARADC_ICNCMR*n*)

The normal conversion mask registers to channel association is described below.

Table 40-5. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register Bits 0:31
ICNCMR0	NCE_CH[31:0]
ICNCMR1	NCE_CH[63:32]
ICNCMR2	NCE_CH[95:64]

Address: 1000h base + A4h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ICNCMR*n* field descriptions

Field	Description
0–31 NCE_CHx	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

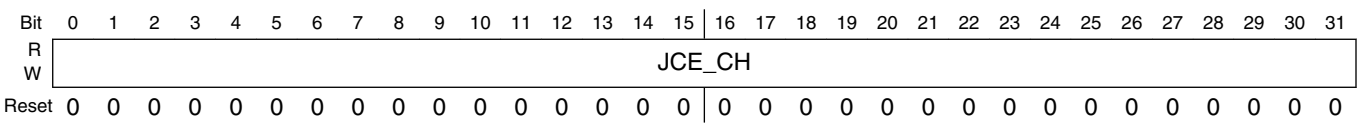
40.5.14 Internal Channel Injected Conversion Mask Register (SARADC_ICJCMR*n*)

The injected conversion mask registers to channel association is described below.

Table 40-6. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register Bits 0:31
ICJCMR0	JCE_CH[31:0]
ICJCMR1	JCE_CH[63:32]
ICJCMR2	JCE_CH[95:64]

Address: 1000h base + B4h offset + (4d × i), where i=0d to 2d

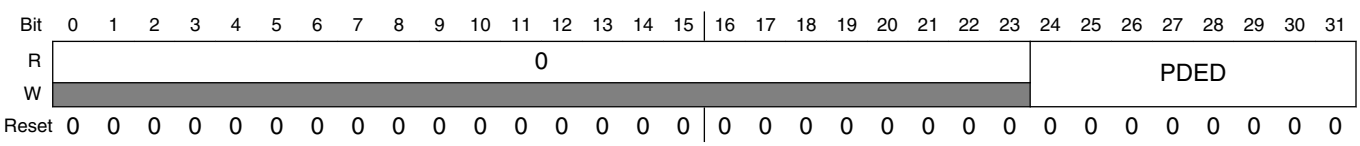


SARADC_ICJCMR*n* field descriptions

Field	Description
0–31 JCE_CH	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

40.5.15 Power Down Exit Delay Register (SARADC_PDEDR)

Address: 1000h base + C8h offset = 10C8h



SARADC_PDEDR field descriptions

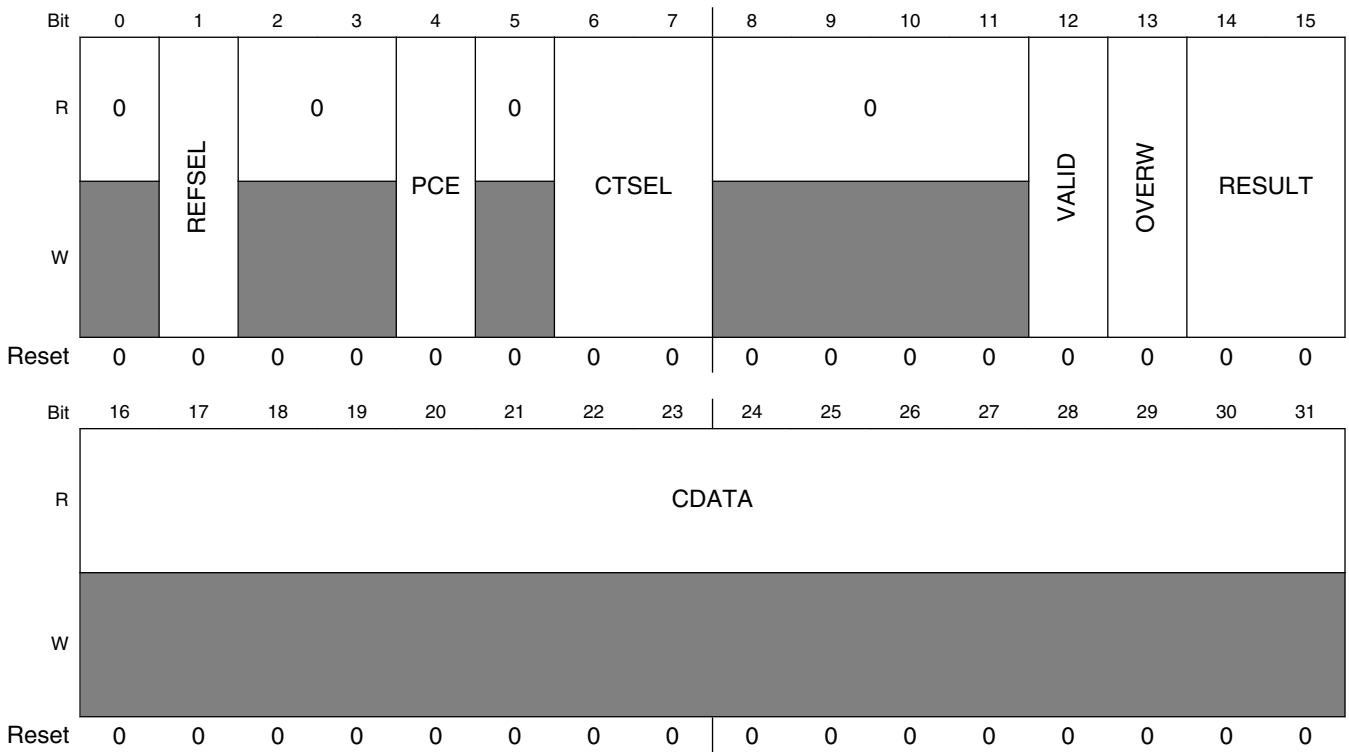
Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PDED	Power down exit delay duration. Defines the delay between the power-down bit reset and the starting of conversion. The power-down delay is calculated as [PDED x 1/(SARADC clock frequency)].

40.5.16 Internal Channel Data Register (SARADC_ICDR*n*)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

This register should be accessed 32-bit R/W.

Address: 1000h base + 100h offset + (4d × i), where i=0d to 95d



SARADC_ICDR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (e.g., : 5 V) 1 Selects the alternate reference (e.g., : 2 V)
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion.

Table continues on the next page...

SARADC_ICDR*n* field descriptions (continued)

Field	Description
	0 Precharge phase is disabled 1 Precharge phase is enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: • When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. • When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14–15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–31 CDATA	Channel converted data **Note:** It is important to note that the content of the CDATA field is affected by the setting of the SARADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When SARADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When SARADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

40.5.17 Internal Channel Watchdog Selection Register (SARADC_ICWSELR*n*)

These registers contain WSEL_CHx[3:0] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described below.

Table 40-7. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register Bits 0:31
ICWSELRx (x = 0 to 11)	WSEL_CH[(8*x)+7] .. WSEL_CH[8*x]

Address: 1000h base + 2B0h offset + (4d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4				WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ICWSELR*n* field descriptions

Field	Description
0–3 WSEL_CH7	Watchdog select for channel 7 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
4–7 WSEL_CH6	Watchdog select for channel 6 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
8–11 WSEL_CH5	Watchdog select for channel 5 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
12–15 WSEL_CH4	Watchdog select for channel 4 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected

Table continues on the next page...

SARADC_ICWSELR*n* field descriptions (continued)

Field	Description
	1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
16–19 WSEL_CH3	Watchdog select for channel 3 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
20–23 WSEL_CH2	Watchdog select for channel 2 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
24–27 WSEL_CH1	Watchdog select for channel 1 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
28–31 WSEL_CH0	Watchdog select for channel 0 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected

40.5.18 Internal Channel Watchdog Enable Register (SARADC_ICWENR*n*)

The Watchdog enable registers to channel association is described below.

Table 40-8. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register Bits 0:31
ICWER0	WEN_CH[31:0]
ICWER1	WEN_CH[63:32]
ICWER2	WEN_CH[95:64]

Address: 1000h base + 2E0h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ICWENR*n* field descriptions

Field	Description
0–31 WEN_CHx	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

40.5.19 Internal Channel Analog Watchdog Out of Range register (SARADC_ICAWORR*n*)

NOTE

These registers are only available in ADCs that have analog watchdogs implemented.

The analog watchdog registers to channel association is described below.

Table 40-9. Internal Channel Analog Watchdog Registers to Channel Association

Register	Register Bits 0:31
ICAWORR0	AWOR_CH[31:0]
ICAWORR1	AWOR_CH[63:32]
ICAWORR2	AWOR_CH[95:64]

Address: 1000h base + 2F0h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CHx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ICAWORR*n* field descriptions

Field	Description
0–31 AWOR_CHx	AWOR_CH[x]: Analog watchdog out of range status for channel x 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

40.5.20 Test Channel Interrupt Pending Register (SARADC_TCIPR)

The interrupt channel register to channel association is described below.

Table 40-10. TCIPR Register to Channel Association

Register	Register Bits0:31
TCIPR	EOC_CH[127:96]

Address: 1000h base + 400h offset = 1400h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CHx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCIPR field descriptions

Field	Description
0–31 EOC_CHx	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

40.5.21 Test Channel Interrupt Mask Register (SARADC_TCIMR)

The interrupt mask register to channel association is described below.

Table 40-11. Test Channel Interrupt Mask Register to Channel Association

Register	Register Bits 0:31
TCIMR	IM_CH[127:96]

Address: 1000h base + 404h offset = 1404h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCIMR field descriptions

Field	Description
0–31 IM_CHx	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

40.5.22 Test Channel DMA Select Register (SARADC_TCDSR)

The DMA select registers to channel association is described below.

Table 40-12. Test Channel DMA Select Registers to Channel Association

Register	Register Bits 0:31
TCDSR	DS_CH[127:96]

Address: 1000h base + 408h offset = 1408h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCDSR field descriptions

Field	Description
0–31 DS_CHx	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

40.5.23 Test Channel Normal Conversion Mask Register (SARADC_TCNCMR)

The normal conversion mask register to channel association is described below.

Table 40-13. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register Bits 0:31
TCNCMR	NCE_CH[127:96]

Address: 1000h base + 40Ch offset = 140Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCNCMR field descriptions

Field	Description
0–31 NCE_CHx	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

40.5.24 Test Channel Injected Conversion Mask Register (SARADC_TCJCMR)

The injected conversion mask register to channel association is described below.

Table 40-14. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register Bits 0:31
TCJCMR	JCE_CH[127:96]

Address: 1000h base + 410h offset = 1410h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCJCMR field descriptions

Field	Description
0–31 JCE_CHx	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

40.5.25 Test Channel Watchdog Selection Register (SARADC_TCWSELR*n*)

This register contains WSEL_CHx[3:0] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select register to channel association is described below.

Table 40-15. Test Channel Watchdog Select Registers to Channel Association

Register	Register Bits 0:31
TCWSELR0	WSEL_CH[103] .. WSEL_CH[96]
TCWSELR1	WSEL_CH[111] .. WSEL_CH[104]
TCWSELR2	WSEL_CH[119] .. WSEL_CH[112]
TCWSELR3	WSEL_CH[127] .. WSEL_CH[120]

Address: 1000h base + 414h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CHa				WSEL_CHb				WSEL_CHc				WSEL_CHd				WSEL_CHe				WSEL_CHf				WSEL_CHg				WSEL_CHh			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCWSELR*n* field descriptions

Field	Description
0–3 WSEL_CHa	Watchdog select for channel a 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
4–7 WSEL_CHb	Watchdog select for channel b 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
8–11 WSEL_CHc	Watchdog select for channel c 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
12–15 WSEL_CHd	Watchdog select for channel d 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
16–19 WSEL_CHe	Watchdog select for channel e 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
20–23 WSEL_CHf	Watchdog select for channel f 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
24–27 WSEL_CHg	Watchdog select for channel g 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected

Table continues on the next page...

SARADC_TCWSELR*n* field descriptions (continued)

Field	Description
	1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
28–31 WSEL_CHh	Watchdog select for channel h 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected

40.5.26 Test Channel Watchdog Enable Register (SARADC_TCWENR)

The Watchdog enable registers to channel association is described below.

Table 40-16. Test Channel Watchdog Enable Registers to Channel Association

Register	Register Bits 0:31
TCWENR	WEN_CH[127:96]

Address: 1000h base + 424h offset = 1424h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCWENR field descriptions

Field	Description
0–31 WEN_CHx	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

40.5.27 Test Channel Analog Watchdog Out of Range Register (SARADC_TCAWORR)

The analog watchdog registers to channel association is described below.

Table 40-17. Analog Watchdog Register to Channel Association

Register	Register Bits 0:31
TCAWORR	AWOR_CH[127:96]

Address: 1000h base + 428h offset = 1428h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CHx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_TCAWORR field descriptions

Field	Description
0–31 AWOR_CHx	AWOR_CH[x]: Analog watchdog out of range status for channel x 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

40.5.28 Test Channel Connection with Analog Pin Register (SARADC_TCCAPR*n*)

Each test channel can be shorted with any internal analog channel using the configuration programmed in these registers.

Test channel short registers to channel association is described below.

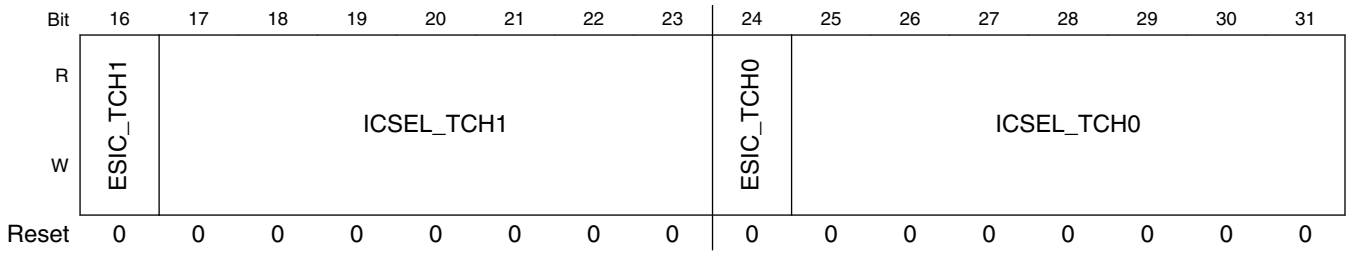
For analog test channel assignment, refer to the chip configuration or ADC configuration section.

Table 40-18. Test Channel Short Registers to Channel Association

Register	Register Bits 0:31
TCCAPRx (x = 0 to 7)	ESIC_TCH[4*x+3], ICSEL_TCH[4*x+3], ESIC_TCH[4*x+2], ICSEL_TCH[4*x+2], ESIC_TCH[4*x+1], ICSEL_TCH[4*x+1], ESIC_TCH[4*x], ICSEL_TCH[4*x]

Address: 1000h base + 430h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESIC_TCH3	ICSEL_TCH3							ESIC_TCH2	ICSEL_TCH2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



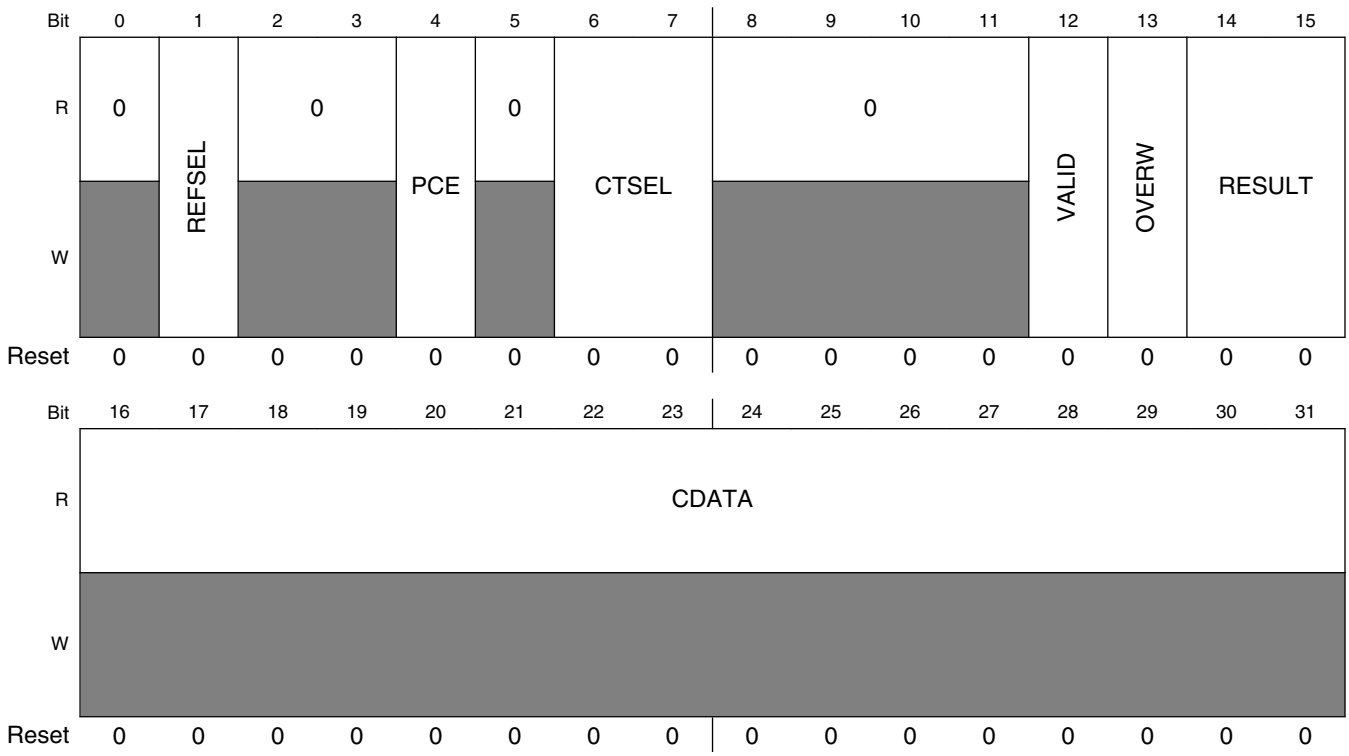
SARADC_TCCAPR*n* field descriptions

Field	Description
0 ESIC_TCH3	Enable short with internal channel for test channel 3 0 Test channel x cannot be shorted with internal channel 1 Test channel x can be shorted with internal channel
1–7 ICSEL_TCH3	Internal channel selection for short with test channel 3
8 ESIC_TCH2	Enable short with internal channel for test channel 2 0 Test channel x cannot be shorted with internal channel 1 Test channel x can be shorted with internal channel
9–15 ICSEL_TCH2	Internal channel selection for short with test channel 2
16 ESIC_TCH1	Enable short with internal channel for test channel 1 0 Test channel x cannot be shorted with internal channel 1 Test channel x can be shorted with internal channel
17–23 ICSEL_TCH1	Internal channel selection for short with test channel 1
24 ESIC_TCH0	Enable short with internal channel for test channel 0 0 Test channel x cannot be shorted with internal channel 1 Test channel x can be shorted with internal channel
25–31 ICSEL_TCH0	Internal channel selection for short with test channel 0

40.5.29 Test Channel Data Register (SARADC_TCDR*n*)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

Address: 1000h base + 450h offset + (4d × i), where i=0d to 31d



SARADC_TCDR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (for e.g., : 5V) 1 Selects the alternate reference (for e.g., : 2V)
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion.

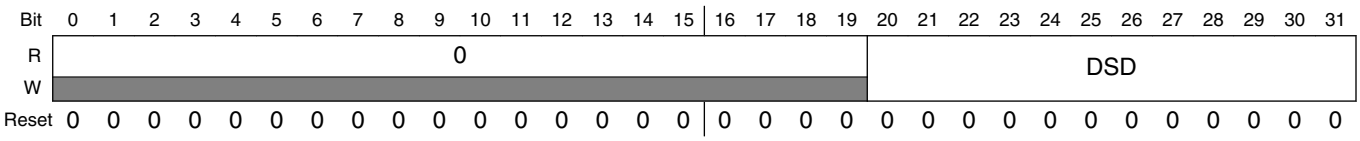
Table continues on the next page...

SARADC_TCDR*n* field descriptions (continued)

Field	Description
	0 Precharge phase is disabled 1 Precharge phase is enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: • When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. • When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14–15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–31 CDATA	Channel converted data **Note:** It is important to note that the content of the CDATA field is affected by the setting of the SARADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When SARADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When SARADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

40.5.30 External Channel Decode Signals Delay Register (SARADC_ECDSDR)

Address: 1000h base + 500h offset = 1500h



SARADC_ECDSDR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DSD	Decode signals delay This bitfield defines the delay between the external decode signals and the start of the sampling phase. It is used to take into account of the settling time required for the external mux. The decode signal delay is calculated as (DSD X 1/Frequency of SARADC clock).

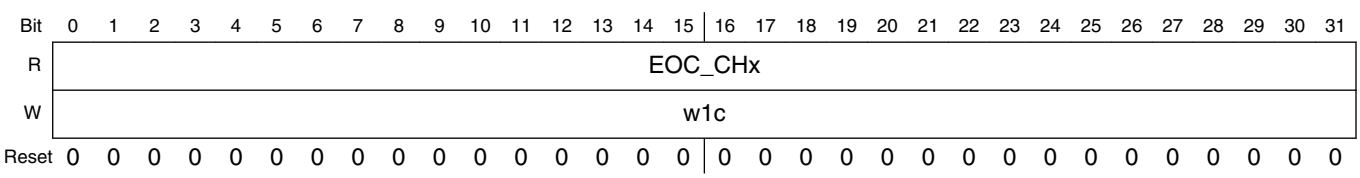
40.5.31 External Channel Interrupt Pending Register (SARADC_ECIPR*n*)

The interrupt channel register to channel association is described below.

Table 40-19. ECIPR0-3 Registers to Channel Association

Register	Register Bits 0:31
ECIPR0	EOC_CH[159:128]
ECIPR1	EOC_CH[191:160]
ECIPR2	EOC_CH[223:192]
ECIPR3	EOC_CH[255:224]

Address: 1000h base + 510h offset + (4d × i), where i=0d to 3d



SARADC_ECIPR*n* field descriptions

Field	Description
0–31 EOC_CHx	End of conversion interrupt pending bit for channel x

SARADC_ECIPR*n* field descriptions (continued)

Field	Description
	0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

40.5.32 External Channel Interrupt Mask Register (SARADC_ECIMR*n*)

The interrupt mask register to channel association is described below.

Table 40-20. External Channel Interrupt Mask Registers to Channel Association

Register	Register Bits 0:31
ECIMR0	IM_CH[159:128]
ECIMR1	IM_CH[191:160]
ECIMR2	IM_CH[223:192]
ECIMR3	IM_CH[255:224]

Address: 1000h base + 520h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECIMR*n* field descriptions

Field	Description
0–31 IM_CHx	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

40.5.33 External Channel DMA Select Register (SARADC_ECDSR*n*)

The DMA select registers to channel association is described below.

Table 40-21. DMA Select Registers to Channel Association

Register	Register Bits 0:31
ECDSR0	DS_CH[159:128]
ECDSR1	DS_CH[191:160]
ECDSR2	DS_CH[223:192]
ECDSR3	DS_CH[255:224]

Address: 1000h base + 530h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECDSR*n* field descriptions

Field	Description
0–31 DS_CHx	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

40.5.34 External Channel Normal Conversion Mask Register (SARADC_ECNCMR*n*)

The normal conversion mask registers to channel association is described below.

Table 40-22. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register Bits 0:31
ECNCMR0	NCE_CH[159:128]
ECNCMR1	NCE_CH[191:160]
ECNCMR2	NCE_CH[223:192]
ECNCMR3	NCE_CH[255:224]

Address: 1000h base + 540h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECNCMR*n* field descriptions

Field	Description
0–31 NCE_CHx	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

40.5.35 External Channel Injected Conversion Mask Register (SARADC_ECJCMR*n*)

The injected conversion mask registers to channel association is described below.

Table 40-23. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register Bits 0:31
ECJCMR0	JCE_CH[159:128]
ECJCMR1	JCE_CH[191:160]
ECJCMR2	JCE_CH[223:192]
ECJCMR3	JCE_CH[255:224]

Address: 1000h base + 550h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECJCMR*n* field descriptions

Field	Description
0–31 JCE_CHx	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

40.5.36 External Channel Watchdog Selection Register (SARADC_ECWSELR*n*)

These registers contain WSEL_CHx[3:0] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described below.

Table 40-24. External Channel Watchdog Enable Registers to Channel Association

Register	Register Bits 0:31
ECWSELRx (x = 0 to 15)	WSEL_CH[(8*x)+135] .. WSEL_CH[(8*x)+128]]

Address: 1000h base + 560h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CHa				WSEL_CHb				WSEL_CHc				WSEL_CHd				WSEL_CHe				WSEL_CHf				WSEL_CHg				WSEL_CHh			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECWSELR*n* field descriptions

Field	Description
0–3 WSEL_CHa	Watchdog select for channel a

Table continues on the next page...

SARADC_ECWSELR*n* field descriptions (continued)

Field	Description
	0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
4–7 WSEL_CHb	Watchdog select for channel b 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
8–11 WSEL_CHc	Watchdog select for channel c 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
12–15 WSEL_CHd	Watchdog select for channel d 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
16–19 WSEL_CHe	Watchdog select for channel e 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
20–23 WSEL_CHf	Watchdog select for channel f 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected
24–27 WSEL_CHg	Watchdog select for channel g 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected

Table continues on the next page...

SARADC_ECWSELR*n* field descriptions (continued)

Field	Description
28–31 WSEL_CHh	Watchdog select for channel h 0000 THRHLR0 register is selected 0001 THRHLR1 register is selected 0010 THRHLR2 register is selected 1110 THRHLR14 register is selected 1111 THRHLR15 register is selected

40.5.37 External Channel Watchdog Enable Register (SARADC_ECWENR*n*)

The Watchdog enable registers to channel association is described below.

Table 40-25. External Channel Watchdog Enable Registers to Channel Association

Register	Register Bits 0:31
ECWER0	WEN_CH[159:128]
ECWER1	WEN_CH[191:160]
ECWER2	WEN_CH[223:192]
ECWER3	WEN_CH[255:224]

Address: 1000h base + 5A0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CHx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECWENR*n* field descriptions

Field	Description
0–31 WEN_CHx	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

40.5.38 External Channel Analog Watchdog Out of Range register (SARADC_ECAWORR*n*)

The analog watchdog registers to channel association is described below.

Table 40-26. External Channel Analog Watchdog Registers to Channel Association

Register	Register Bits 31:0
ECAWORR0	AWOR_CH[159:128]
ECAWORR1	AWOR_CH[191:160]
ECAWORR2	AWOR_CH[223:192]
ECAWORR3	AWOR_CH[255:224]

Address: 1000h base + 5B0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CHx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECAWORR*n* field descriptions

Field	Description
0–31 AWOR_CHx	AWOR_CH[x]: Analog watchdog out of range status for channel x 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

40.5.39 External Channel Mapping to Internal Channel Register (SARADC_ECMICR*n*)

Each set of 8 external channels can be mapped with any internal analog channel using the configuration programmed in these registers.

Table 40-27. Test Channel Short Registers to Channel Association

Register	Register Bits 0:31
ECMICRx (x = 0 to 3)	ICSEL_ECH[128 + (32*x+31)]_[128 + (32*x+24)] ICSEL_ECH[128 + (32*x + 23)]_ [128 + (32*x+16)], ICSEL_ECH[128 + (32*x+15)]_[128 + (32*x+8)] ICSEL_ECH[128 + (32*x + 7)]_ [128 + 32*x],

Address: 1000h base + 5C0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	ICSEL_ECH152_159							Reserved	ICSEL_ECH144_151						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	ICSEL_ECH136_143							Reserved	ICSEL_ECH128_135						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SARADC_ECMICR*n* field descriptions

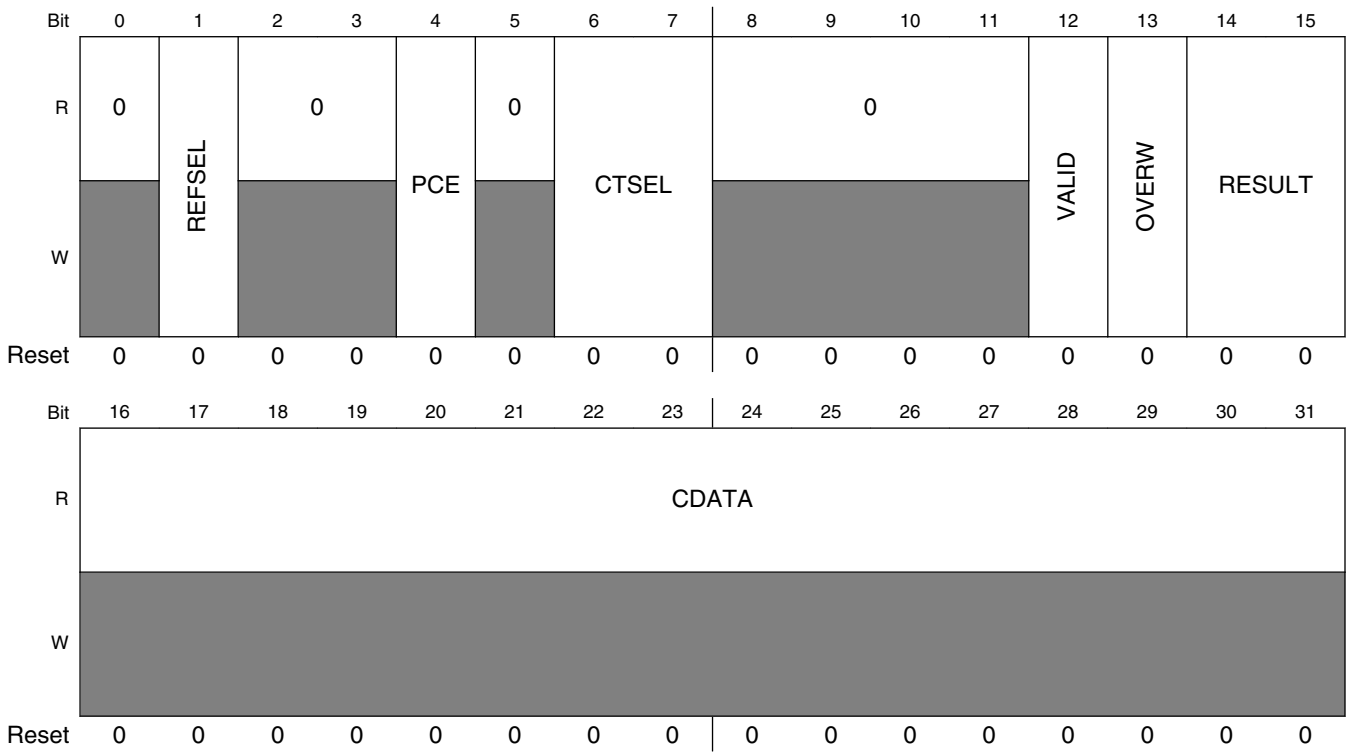
Field	Description
0 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
1–7 ICSEL_ECH152_159	Internal channel selection for external channels (128 + 8x) to (135+8*x) where x = 0,1,2..16
8 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
9–15 ICSEL_ECH144_151	Internal channel selection for external channels (128 + 8x) to (135+8*x) where x = 0,1,2..16
16 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
17–23 ICSEL_ECH136_143	Internal channel selection for external channels (128 + 8x) to (135+8*x) where x = 0,1,2..16
24 Reserved	This field is reserved. Reserved Write of any value has no effect; read value is always 0.
25–31 ICSEL_ECH128_135	Internal channel selection for external channels (128 + 8x) to (135+8*x) where x = 0,1,2..16

40.5.40 External Channel Data Register (SARADC_ECDR*n*)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

This register should be accessed 32-bit R/W.

Address: 1000h base + 5D0h offset + (4d × i), where i=0d to 127d



SARADC_ECDR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (for e.g., : 5 V) 1 Selects the alternate reference (for e.g., : 2 V)
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected

Table continues on the next page...

SARADC_ECDR*n* field descriptions (continued)

Field	Description
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: • When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. • When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14–15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–31 CDATA	Channel converted data **Note:** It is important to note that the content of the CDATA field is affected by the setting of the SARADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When SARADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When SARADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

40.6 Start conversion pulse delay

The following table lists delays between various trigger point (initiation request for conversion) and actual issue of start pulse (adc_start) to the ADC hardmacrocell. Here each possible trigger source is described with the number of clock cycles required for different operations until trigger of adc_start.

Table 40-28. Start of conversion pulse delay

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
Normal conversion triggered by	1	—	2	1-2[1]	Cumulative Delay from wen(byte_en)

Table continues on the next page...

Table 40-28. Start of conversion pulse delay (continued)

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
software (setting of NSTART of MCR)					assertion to actual issue of adc_start pulse = 3-4 ipg_clk_adc cycles + 1 ipg_clk cycle.
Normal conversion triggered by external trigger (HW trigger through ipp_ind_start_trg port)	—	—	2	1–2[1]	Cumulative delay from the triggering edge (rising or falling) to the actual issue of adc_start pulse = 3–4 ipg_clk_adc cycles.
Normal conversion within a chain	—	—	0 (from EOC of previous conversion)	—	—
1 (from last_evaluation_cyc of previous conversion)	—	Delay from the EOC of last conversion, The start of next conversion is issued exactly on the same cycle when EOC is expected for previous conversion, start pulse in issued looking at last_evalution_cyc_i which is received one cycle prior to EOC.	—	—	—
Injected conversion triggered by software (setting of JSTART bit of MCR)	1	—	3	1–2[1]	Cumulative delay from wen(byte_en) assertion. Here delay for first start pulse of the injected chain execution when FSM was originally in IDLE state is considered = 4–5 ipg_clk_adc cycles + 1 ipg_clk cycle.
Injected conversion triggered by external Trigger (HW trigger through ipp_ind_injection_trg port)	—	—	3	1–2[1]	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 ipg_clk_adc cycles.

Table continues on the next page...

Table 40-28. Start of conversion pulse delay (continued)

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
Injected conversion within a chain (delay between consecutive conversion of injected chain)	—	—	0 (from EOC of previous conversion)	—	—
1 (from last_evaluation_cyc of previous conversion)	—	Delay from EOC of last injected channel and adc_start pulse of next injected channel.	—	—	—
The start of next conversion is issued exactly on the same cycle when EOC is expected for previous conversion, start pulse in issued looking at last_evalution_cyc_i which is received one cycle prior to EOC					
Injected conversion over normal conversion (software injected conversion chain)	1	1–2[1]	3–5[2]	1–2[1]	Cumulative delay from wen(byte_en) assertion to actual trigger of first injected channel, when the FSM was in conversion cycle of previous normal conversion = 4–7 ipg_clk_adc cycles + 2-3 ipg_clk cycles.
Injection of conversion over normal conversion in JTRGSEQ mode	—	—	3	1–2[1]	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 ipg_clk_adc cycles.
Abort of a Normal/ Injected Channel conversion during Normal Chain/ Injected Chain execution (setting	1	1–2[1]	3–5[2]	1–2[1]	Delay from writing of ABORT bit (wen/ byte_en assertion) to actual issue of adc_start pulse for the next channel

Table 40-28. Start of conversion pulse delay

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
of MCR[ABORT] bit)					conversion = 4–7 ipg_clk_adc cycles + 2–3 ipg_clk cycles.

1. The variation is due to inter domain synchronization.
2. The delay values are calculated over a range of system and ADC clock frequencies. With ipg_clk(max) = 50 MHz to ipg_clk(min) = 40 MHz and ipg_clk_adc(max) = 16 MHz to ipg_clk_adc(min) = 14 MHz. L.H.S value corresponds to max frequencies and RHS corresponds to min frequencies.

Chapter 41
Temperature Sensor

41.1 Introduction

The device includes an on-chip temperature sensor that monitors device temperature and delivers one analog output signal and three digital output signals. The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device. The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage, sampled bandgap reference voltage, which comes from another module, and calibration parameter values stored in internal flash memory. The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions.

The digital outputs signal when the junction temperature drifts below the low temperature threshold or above one of two high temperature thresholds. These signals notify the device to take action to appropriately adjust the device temperature in response to an out of specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

41.2 Functional Description

The temperature sensor generates one analog output voltage, $V_{TSENS}(T)$, which is proportional to the absolute current junction temperature of the device, and three digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or one of two pre-set high temperature thresholds.

An on-chip ADC module is used to convert the analog output voltage, as well as the bandgap reference voltage, into a digital representation. These values, along with parameter values stored in on-chip flash memory, are used by software to calculate the device junction temperature.

41.2.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in on-chip flash memory. While enabled, if a new trimming value is loaded into the Temperature Sensor module, it may take up to 1 microsecond for the digital outputs to be updated to reflect the new trimming condition.

When the temperature threshold detection feature is enabled , the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the following temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold (-40°C).
- The high temperature digital output signals if the junction temperature rises above the first high temperature threshold (150°C).

The threshold detection circuit is calibrated with respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to –40°C), and the high temperature thresholds are trimmed such that one of them matches the hot insertion test temperature (nominally equal to 150°C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated on Figure 41-1.

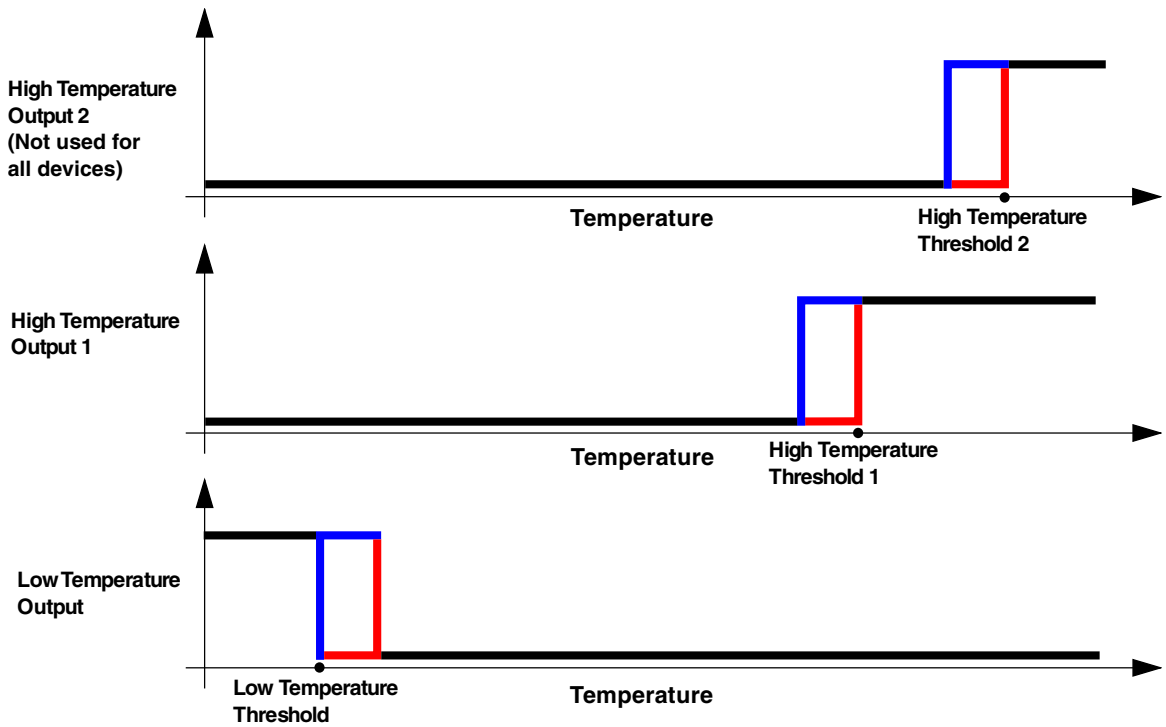


Figure 41-1. Digital output behavior with temperature

41.2.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

41.3 Temperature formula

The system chain that translates device junction temperature into a digital variable is composed of the temperature sensor, a bandgap reference voltage source and the on-chip ADC. Both analog output voltages of the temperature sensor and the bandgap reference voltage source must be converted by the ADC into digital codes to obtain the device junction temperature. The temperature formula shown in Figure 41-2 is used to calculate internal device junction temperature.

$$T = (232 + TSCA \times 2^{-6}) \times \frac{V_{TSENS\_CODE}(T)}{V_{BG\_CODE}(T)} - (273 + TSCB \times 2^{-4})\ [^{o}C]$$

Where:

$$TSCA = \left(\frac{T_{HIGH} - T_{LOW}}{V_{TSENS\_CODE}(T_{HIGH}) - V_{TSENS\_CODE}(T_{LOW})} \times V_{BG\_CODE}(T_{FINAL}) - 232 \right) \times 2^{6}$$

$$TSCB = \left(\frac{T_{HIGH} - T_{LOW}}{V_{TSENS\_CODE}(T_{HIGH}) - V_{TSENS\_CODE}(T_{LOW})} \times V_{TSENS\_CODE}(T_{LOW}) - T_{LOW} - 273 \right) \times 2^{4}$$

$$V_{TSENS\_CODE}(T_{LOW}) = \frac{V_{TSENS}(T_{LOW})}{V_{ref0}} \times 2^{12}$$

$$V_{TSENS\_CODE}(T_{HIGH}) = \frac{V_{TSENS}(T_{HIGH})}{V_{ref0}} \times 2^{12}$$

$$V_{BG\_CODE}(T_{FINAL}) = \frac{V_{BG}(T_{FINAL})}{V_{ref0}} \times 2^{12}$$

$$V_{BG\_CODE}(T) = \frac{V_{BG}(T)}{V_{ref}} \times 2^{12}$$

$$V_{TSENS\_CODE}(T) = \frac{V_{TSENS}(T)}{V_{ref}} \times 2^{12}$$

Notes:

- TSCA and TSCB are 16-bit signed integer constants stored in flash memory during factory test and calibration
- $V_{TSENS}(T)$ is the temperature sensor output sampled by the ADC
- $V_{BG}(T)$ is the bandgap reference voltage sampled by the ADC at temperature T
- V_{ref} is the ADC reference voltage in normal usage of the device
- V_{ref0} is the ADC reference voltage during factory calibration
- T_{LOW} is the low temperature factory calibration temperature
- T_{HIGH} is the hot temperature factory calibration temperature
- T_{FINAL} is the final insertion factory calibration temperature

Figure 41-2. Temperature formula

Parameters TSCA and TSCB are 16-bit signed integer constants stored during factory test and calibration that need to be used in the temperature formula to calculate device junction temperature. The values for TSCA and TSCB translate into positive or negative shifts of the linear relation coefficients that compensate for manufacturing process variation. Figure 41-2 shows how parameters TSCA and TSCB are calculated in terms of the actual variables measured during factory calibration.

41.4 Calculating device temperature

Software must perform the following steps in order to calculate device temperature.

1. Measure the bandgap reference voltage using the on-chip ADC module and calculate $V_{BG\_CODE}(T)$ according to the formula shown in Figure 41-2.
2. Measure the temperature sensor output voltage using the on-chip ADC module and calculate $V_{TSENS\_CODE}(T)$ according to the formula shown in Figure 41-2
3. Retrieve parameters TSCA and TSCB from flash memory.
4. Calculate T according to the formula shown in Figure 41-2.

Note

Temperature Formula may be calculated using signed 32-bit (integer) operations showing negligible precision loss.

Chapter 42
System Timer Module (STM)

42.1 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

42.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

42.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

42.1.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

42.2 External signal description

The STM does not have any external interface signals.

42.3 Memory map and registers

The STM programming model includes a group of 32-bit registers—a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

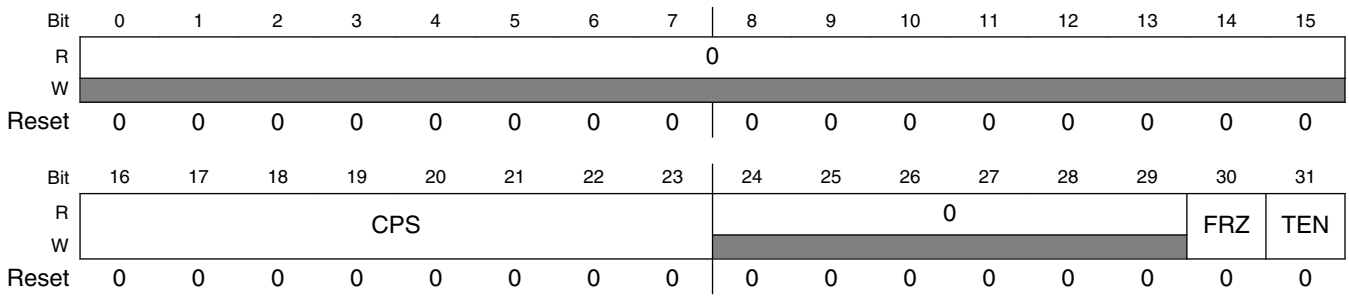
STM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STM Control Register (STM_CR)	32	R/W	0000_0000h	42.3.1/1675
4	STM Count Register (STM_CNT)	32	R/W	0000_0000h	42.3.2/1676
10	STM Channel Control Register (STM_CCR0)	32	R/W	0000_0000h	42.3.3/1676
14	STM Channel Interrupt Register (STM_CIR0)	32	w1c	0000_0000h	42.3.4/1677
18	STM Channel Compare Register (STM_CMP0)	32	R/W	0000_0000h	42.3.5/1677
20	STM Channel Control Register (STM_CCR1)	32	R/W	0000_0000h	42.3.3/1676
24	STM Channel Interrupt Register (STM_CIR1)	32	w1c	0000_0000h	42.3.4/1677
28	STM Channel Compare Register (STM_CMP1)	32	R/W	0000_0000h	42.3.5/1677
30	STM Channel Control Register (STM_CCR2)	32	R/W	0000_0000h	42.3.3/1676
34	STM Channel Interrupt Register (STM_CIR2)	32	w1c	0000_0000h	42.3.4/1677
38	STM Channel Compare Register (STM_CMP2)	32	R/W	0000_0000h	42.3.5/1677
40	STM Channel Control Register (STM_CCR3)	32	R/W	0000_0000h	42.3.3/1676
44	STM Channel Interrupt Register (STM_CIR3)	32	w1c	0000_0000h	42.3.4/1677
48	STM Channel Compare Register (STM_CMP3)	32	R/W	0000_0000h	42.3.5/1677

42.3.1 STM Control Register (STM_CR)

The STM Control Register includes the prescale value, freeze control and timer enable bits.

Address: 0h base + 0h offset = 0h



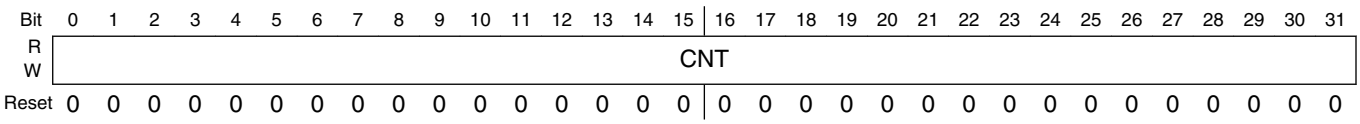
STM_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ---------- 0xFF Divide system clock by 256
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **NOTE:** When the MCU enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

42.3.2 STM Count Register (STM_CNT)

The STM Count Register holds the timer count value.

Address: 0h base + 4h offset = 4h



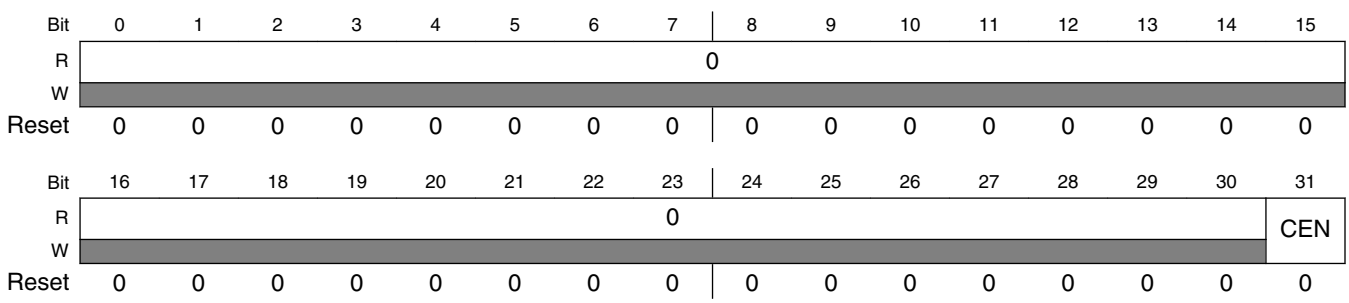
STM_CNT field descriptions

Field	Description
0–31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

42.3.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Address: 0h base + 10h offset + (16d × i), where i=0d to 3d



STM_CCR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

42.3.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register has the interrupt flag for channel n of the timer.

Address: 0h base + 14h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CIR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

42.3.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Address: 0h base + 18h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CMP*n* field descriptions

Field	Description
0–31 CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

STM_CMP*n* field descriptions (continued)

Field	Description

42.4 Functional description

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode, the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode; otherwise, it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*), and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

NOTE

The STM counter does not advance when the system clock is stopped.

Chapter 43
Software Watchdog Timer (SWT)

43.1 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

43.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out.

43.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

43.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

43.2 External signal description

The SWT module does not have any external interface signals.

43.3 Memory Map and Registers

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in the following table.

SWT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SWT Control Register (SWT_CR)	32	R/W	See section	43.3.1/1681
4	SWT Interrupt Register (SWT_IR)	32	R/W	0000_0000h	43.3.2/1684
8	SWT Time-out Register (SWT_TO)	32	R/W	See section	43.3.3/1684
C	SWT Window Register (SWT_WN)	32	R/W	0000_0000h	43.3.4/1685
10	SWT Service Register (SWT_SR)	32	W	0000_0000h	43.3.5/1685
14	SWT Counter Output Register (SWT_CO)	32	R	0000_0000h	43.3.6/1686
18	SWT Service Key Register (SWT_SK)	32	R/W	0000_0000h	43.3.7/1687

43.3.1 SWT Control Register (SWT_CR)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	Reserved	STP	FRZ	WEN
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

SWT_CR field descriptions

Field	Description
0 MAP0	Master Access Protection for Master 0. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
2 MAP2	Master Access Protection for Master 2. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
3 MAP3	Master Access Protection for Master 3. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode. 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudorandom key values to the SWT_SR. 10 Fixed Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register, which cannot be updated while the watchdog is enabled. 11 Incremental Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register, which can be updated.
23 RIA	Reset on Invalid Access. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
24 WND	Window Mode.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
	0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25 ITR	Interrupt Then Reset. 0 Generate a reset on a time-out 1 Generate an interrupt on an initial time-out, reset on a second consecutive time-out
26 HLK	Hard Lock. This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock. This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 Reserved	Reserved This field is reserved.
29 STP	Stop Mode Control. Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control. Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled. 0 SWT is disabled 1 SWT is enabled

43.3.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request 1 Interrupt request due to an initial time-out

43.3.3 SWT Time-out Register (SWT_TO)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT Time-out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTO																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

SWT_TO field descriptions

Field	Description
0–31 WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value or 0x100, whichever is greater, when the service sequence is written or when the SWT is enabled.

43.3.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_WN field descriptions

Field	Description
0–31 WST	Window Start Value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

43.3.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																	WSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] field is 01b, two pseudorandom key values are written to service the watchdog, see Functional description for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

43.3.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

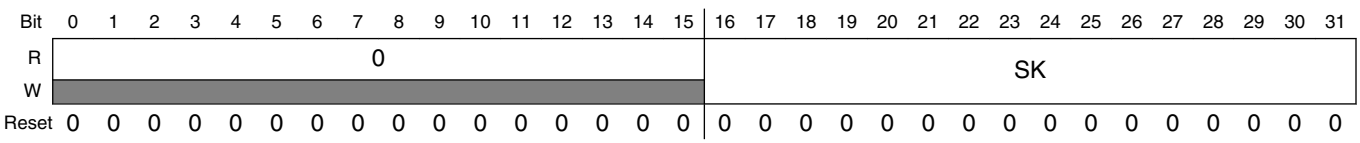
SWT_CO field descriptions

Field	Description
0–31 CNT	Watchdog Count. When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

43.3.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 18h offset = 18h



SWT_SK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] is 01b, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^{16} .

43.4 Functional description

43.4.1 Introduction

The SWT is a 32-bit window watchdog timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- a control register (SWT_CR),
- an interrupt register (SWT_IR),
- a time-out register (SWT_TO),
- a window register (SWT_WN),
- a service register (SWT_SR),
- a counter output register (SWT_CO), and
- a service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

43.4.1.1 SWT_CR register

The SWT_CR register includes bits to enable the timer, set configuration options, and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific.[3] If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

43.4.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100, in which case the time-out period is set to 0x100. When the SWT is enabled, the time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. See the Configuration section to determine the source that is used to clock the down counter.

43.4.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

3. See the chip-specific SWT information.

43.4.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

43.4.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

43.4.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

43.4.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

43.4.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation that is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence
- fixed execution address
- incremental execution address

43.4.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected, which requires writing 0xA602, then 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

43.4.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected, which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined by the equation in the following figure. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 \times SK_n + 3) \bmod 2^{16}$$

Figure 43-1. Pseudorandom Key Generator

43.4.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register cannot be updated while the watchdog is enabled.

43.4.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

43.4.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000, then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

43.4.5 Time-out

The interrupt-then-reset bit (SWT_CR[ITR]) controls the action taken when a time-out occurs. If the SWT_CR[ITR] bit is not set, a reset is generated immediately on a time-out. If the SWT_CR[ITR] bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR[TIF]). The interrupt request is cleared by writing a one to the SWT_IR[TIF] bit.

43.4.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

Chapter 44
Periodic Interrupt Timer (PIT)

44.1 Introduction

The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels. It includes a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup.

Note

For the chip-specific implementation details of this module's instances see the Device Configuration chapter.

44.1.1 Block diagram

The following figure shows the PIT block diagram.

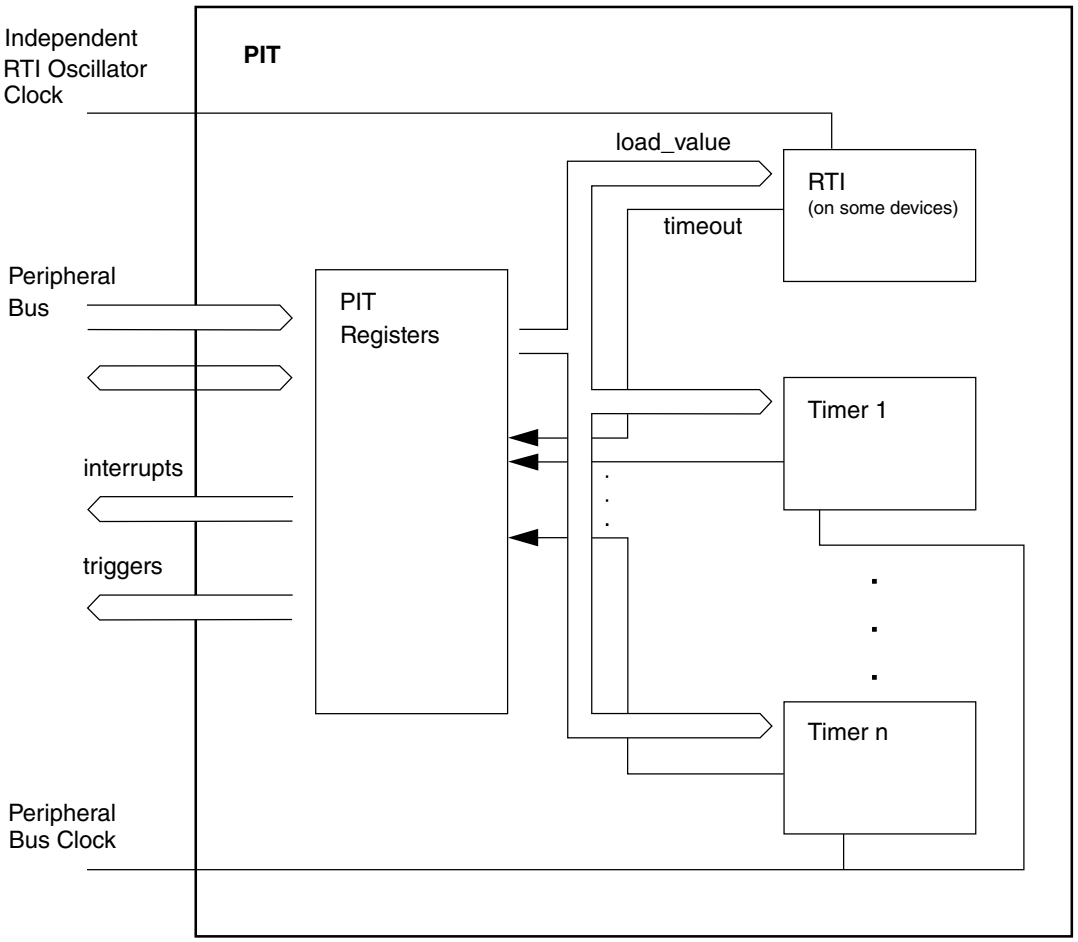


Figure 44-1. Block diagram of the PIT

Note

Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

44.1.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wake up the CPU in Stop mode
- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- RTI interrupt can be raised, even when the bus clock is switched off

- Power saving with a separate input clock for the RTI timer (all other timers share one common core clock)
- Independent timeout periods for each timer

44.2 Signal description

This module has no external pins.

44.3 Memory map/register description

This section provides a detailed description of all registers accessible in this module.

Note

Reserved registers will read as 0, writes will have no effect.

Note

Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

Note

The RTI registers should be programmed only when the RTI clock is running.

PIT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0006h	44.3.1/1697
E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	44.3.2/1698
E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	44.3.3/1698
F0	PIT RTI Timer Load Value Register (PIT_RTI_LDVAL)	32	R/W	0000_0000h	44.3.4/1699
F4	PIT RTI Current Timer Value Register (PIT_RTI_CVAL)	32	R	0000_0000h	44.3.5/1699
F8	PIT RTI Timer Control Register (PIT_RTI_TCTRL)	32	R/W	0000_0000h	44.3.6/1700
FC	PIT RTI Timer Flag Register (PIT_RTI_TFLG)	32	w1c	0000_0000h	44.3.7/1701
100	PIT Timer Load Value Register n (PIT_LDVAL0)	32	R/W	0000_0000h	44.3.8/1702
104	PIT Current Timer Value Register n (PIT_CVAL0)	32	R	0000_0000h	44.3.9/1702

Table continues on the next page...

PIT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
108	PIT Timer Control Register n (PIT_TCTRL0)	32	R/W	0000_0000h	44.3.10/ 1703
10C	PIT Timer Flag Register n (PIT_TFLG0)	32	w1c	0000_0000h	44.3.11/ 1704
110	PIT Timer Load Value Register n (PIT_LDVAL1)	32	R/W	0000_0000h	44.3.8/1702
114	PIT Current Timer Value Register n (PIT_CVAL1)	32	R	0000_0000h	44.3.9/1702
118	PIT Timer Control Register n (PIT_TCTRL1)	32	R/W	0000_0000h	44.3.10/ 1703
11C	PIT Timer Flag Register n (PIT_TFLG1)	32	w1c	0000_0000h	44.3.11/ 1704
120	PIT Timer Load Value Register n (PIT_LDVAL2)	32	R/W	0000_0000h	44.3.8/1702
124	PIT Current Timer Value Register n (PIT_CVAL2)	32	R	0000_0000h	44.3.9/1702
128	PIT Timer Control Register n (PIT_TCTRL2)	32	R/W	0000_0000h	44.3.10/ 1703
12C	PIT Timer Flag Register n (PIT_TFLG2)	32	w1c	0000_0000h	44.3.11/ 1704
130	PIT Timer Load Value Register n (PIT_LDVAL3)	32	R/W	0000_0000h	44.3.8/1702
134	PIT Current Timer Value Register n (PIT_CVAL3)	32	R	0000_0000h	44.3.9/1702
138	PIT Timer Control Register n (PIT_TCTRL3)	32	R/W	0000_0000h	44.3.10/ 1703
13C	PIT Timer Flag Register n (PIT_TFLG3)	32	w1c	0000_0000h	44.3.11/ 1704
140	PIT Timer Load Value Register n (PIT_LDVAL4)	32	R/W	0000_0000h	44.3.8/1702
144	PIT Current Timer Value Register n (PIT_CVAL4)	32	R	0000_0000h	44.3.9/1702
148	PIT Timer Control Register n (PIT_TCTRL4)	32	R/W	0000_0000h	44.3.10/ 1703
14C	PIT Timer Flag Register n (PIT_TFLG4)	32	w1c	0000_0000h	44.3.11/ 1704
150	PIT Timer Load Value Register n (PIT_LDVAL5)	32	R/W	0000_0000h	44.3.8/1702
154	PIT Current Timer Value Register n (PIT_CVAL5)	32	R	0000_0000h	44.3.9/1702
158	PIT Timer Control Register n (PIT_TCTRL5)	32	R/W	0000_0000h	44.3.10/ 1703
15C	PIT Timer Flag Register n (PIT_TFLG5)	32	w1c	0000_0000h	44.3.11/ 1704
160	PIT Timer Load Value Register n (PIT_LDVAL6)	32	R/W	0000_0000h	44.3.8/1702
164	PIT Current Timer Value Register n (PIT_CVAL6)	32	R	0000_0000h	44.3.9/1702
168	PIT Timer Control Register n (PIT_TCTRL6)	32	R/W	0000_0000h	44.3.10/ 1703
16C	PIT Timer Flag Register n (PIT_TFLG6)	32	w1c	0000_0000h	44.3.11/ 1704
170	PIT Timer Load Value Register n (PIT_LDVAL7)	32	R/W	0000_0000h	44.3.8/1702
174	PIT Current Timer Value Register n (PIT_CVAL7)	32	R	0000_0000h	44.3.9/1702

Table continues on the next page...

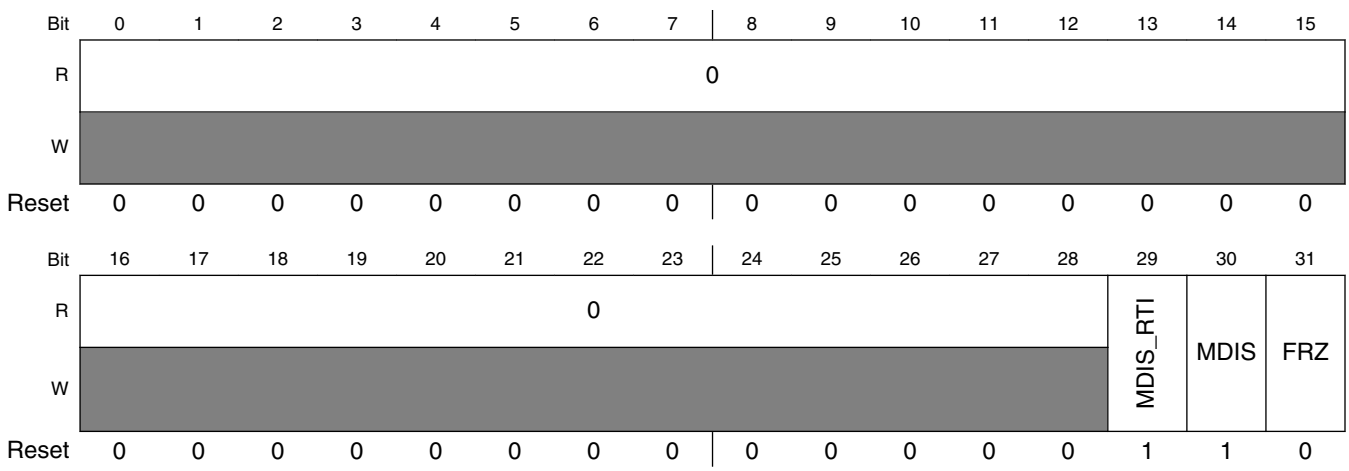
PIT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
178	PIT Timer Control Register n (PIT_TCTRL7)	32	R/W	0000_0000h	44.3.10/ 1703
17C	PIT Timer Flag Register n (PIT_TFLG7)	32	w1c	0000_0000h	44.3.11/ 1704

44.3.1 PIT Module Control Register (PIT_MCR)

The PIT Module Control Register (PIT*x*_MCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Address: FFF8_0000h base + 0h offset = FFF8_0000h



PIT_MCR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI timers is enabled. 1 Clock for RTI timers is disabled.
30 MDIS	Module Disable This is used to disable the module clock. The RTI timer is not affected by this bit. This bit must be enabled before any other setup is done. 0 Clock for PIT timers is enabled. 1 Clock for PIT timers is disabled.

Table continues on the next page...

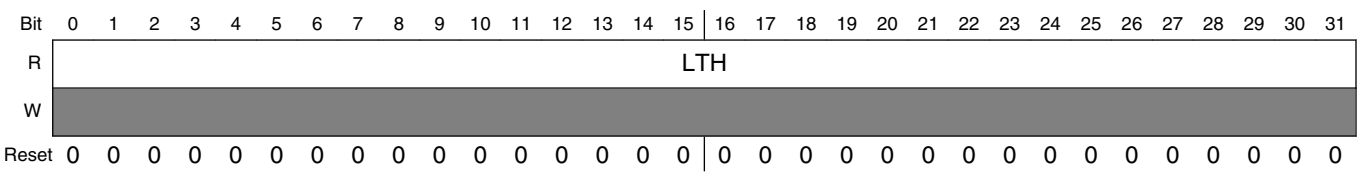
PIT_MCR field descriptions (continued)

Field	Description
31 FRZ	Freeze Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

44.3.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)

The PIT Module *x* Upper Lifetime Timer Register (PIT*x*_LTMR64H) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

Address: FFF8_0000h base + E0h offset = FFF8_00E0h



PIT_LTMR64H field descriptions

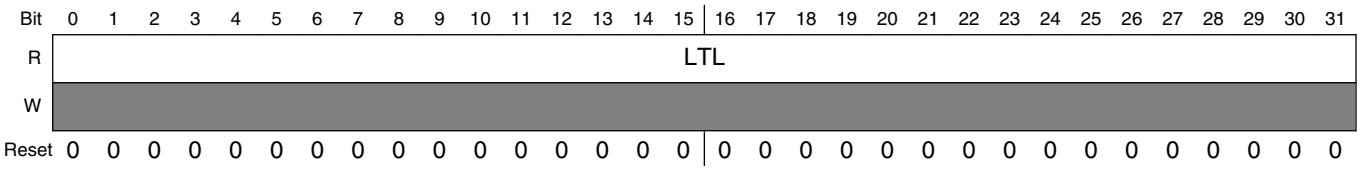
Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

44.3.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)

The PIT Module *x* Lower Lifetime Timer Register (PIT*x*_LTMR64L) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Address: FFF8_0000h base + E4h offset = FFF8_00E4h



PIT_LTMR64L field descriptions

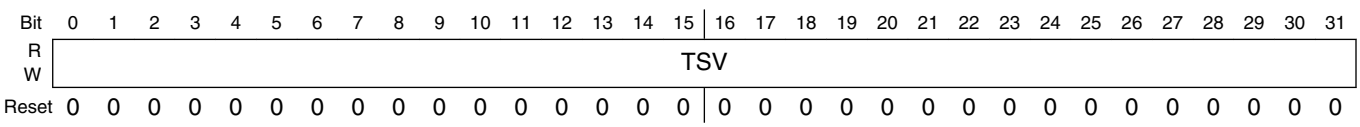
Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

44.3.4 PIT RTI Timer Load Value Register (PIT_RTI_LDVAL)

This register select the timeout period for the timer interrupts.

In the case of the RTI, it takes several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronization mechanism allows 0 wait states in this case.

Address: FFF8_0000h base + F0h offset = FFF8_00F0h



PIT_RTI_LDVAL field descriptions

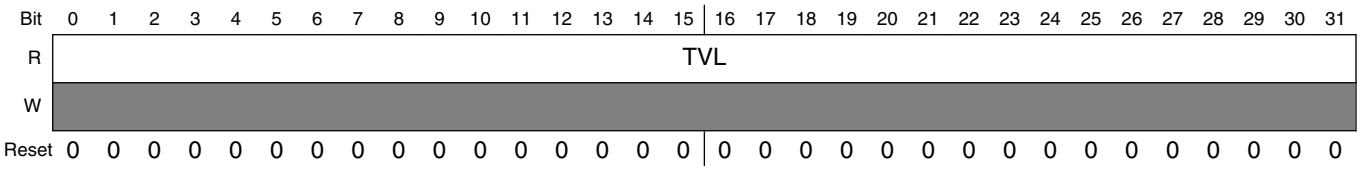
Field	Description
0–31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. **NOTE:** The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

44.3.5 PIT RTI Current Timer Value Register (PIT_RTI_CVAL)

This register indicate the current timer position.

For the RTI timers, it shows a value which is several cycles old, since it originates from a potentially different clock domain.

Address: FFF8_0000h base + F4h offset = FFF8_00F4h



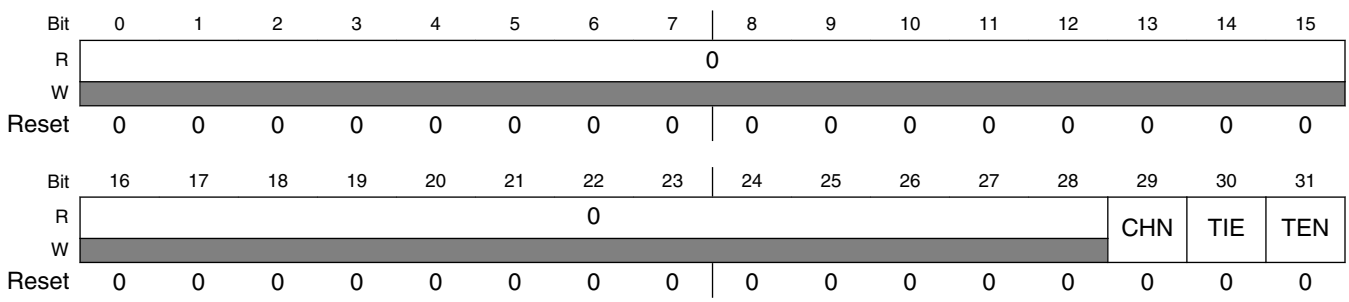
PIT_RTI_CVAL field descriptions

Field	Description
0–31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **NOTE:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

44.3.6 PIT RTI Timer Control Register (PIT_RTI_TCTRL)

This register contain the control bits for each timer.

Address: FFF8_0000h base + F8h offset = FFF8_00F8h



PIT_RTI_TCTRL field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be changed. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer 2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.

Table continues on the next page...

PIT_RTI_TCTRL field descriptions (continued)

Field	Description
	When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

44.3.7 PIT RTI Timer Flag Register (PIT_RTI_TFLG)

This register holds the PIT interrupt flags.

Address: FFF8_0000h base + FCh offset = FFF8_00FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_TFLG field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

44.3.8 PIT Timer Load Value Register n (PIT_LDVAL*n*)

Address: FFF8_0000h base + 100h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LDVAL*n* field descriptions

Field	Description
0–31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

44.3.9 PIT Current Timer Value Register n (PIT_CVAL*n*)

Address: FFF8_0000h base + 104h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_CVAL*n* field descriptions

Field	Description
0–31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **NOTE:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

44.3.10 PIT Timer Control Register n (PIT_TCTRL*n*)

Address: FFF8_0000h base + 108h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be changed. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

44.3.11 PIT Timer Flag Register n (PIT_TFLG*n*)

Address: FFF8_0000h base + 10Ch offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

44.4 Functional description

This section provides the functional description of the module.

44.4.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts. Each interrupt is available on a separate interrupt line. Additionally the RTI timer can be used to wake up the processor.

44.4.1.1 Timers/RTI

The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 should be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see the following figure).

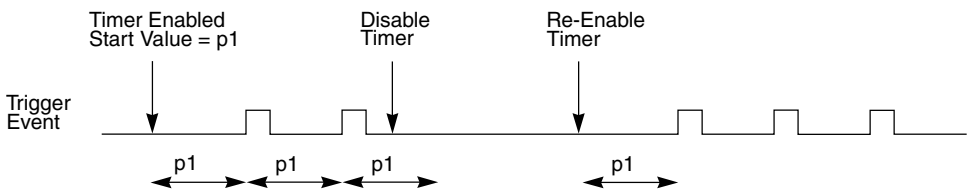


Figure 44-2. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see the following figure).

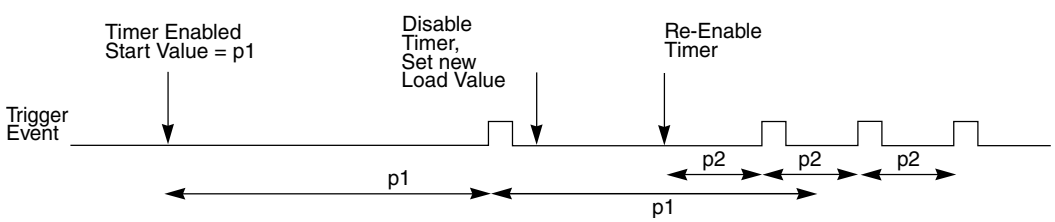


Figure 44-3. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (see the following figure).

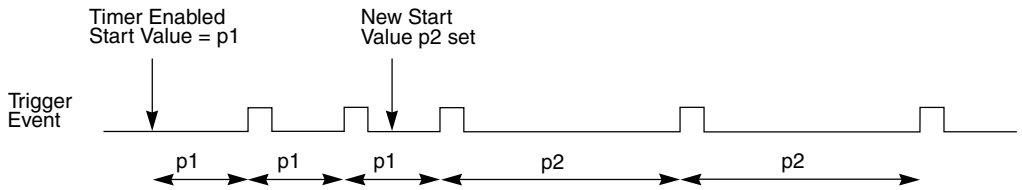


Figure 44-4. Dynamically setting a new load value

44.4.1.2 Debug mode

In debug mode, the timers will be frozen based on field PIT*x*_MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

44.4.2 Interrupts

All of the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting the TIE bits. The timer interrupt flags (TIF) are set to 1 when a timeout occurs on the associated timer, and are cleared to '0' by writing a '1' to that TIF bit.

The PIT RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting the TIE bit to zero. The real time interrupt flag (TIF) is set to '1' when a timeout occurs, and is cleared by writing a '1' to the TIF bit. (The flag will be set regardless whether the interrupt is enabled.)

The RTI can be used for periodic wakeup from a low power mode. It can also be used to generate a general purpose interrupt.

44.4.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer *n*-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

44.5 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.
- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

First the PIT module must be activated by writing a 0 to field PITx_MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every 500 ms/100 ns = 5000000 cycles. Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) –1

This means RTI_LDVAL will be written with 004C4B3F hex, LDVAL1 should be written with 0x0003E7FF, and LDVAL3 should be written with 0x0016E35F.

To generate the wakeup interrupt, the interrupt line must be enabled by writing a '1' to bit RTI_TCTRL[TIE]. To start the RTI, bit RTI_TCTRL[TEN] must also be set.

The interrupt for Timer 1 is enabled by setting bit TCTRL1[TIE]. The timer is started by writing '1' to bit TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a '1' to bit TCTRL3[TEN]; bit TIE stays at 0.

The following example code matches the described setup:

```
                    // turn on PIT
                   PIT_MCR = 0x00;
                      // RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
  PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
           PIT_RTI_TCTRL |= PIT_TEN; // start RTI
                     // Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
       PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
```

```
                PIT_TCTRL1 |= TEN; // start Timer 1
                           // Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
                PIT_TCTRL3 |= TEN; // start Timer 3
```

44.6 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 hour.

First the PIT module needs to be activated by writing a '0' to field PIT*x*_MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to timer 1 and programmed to trigger every 10 times.

The value for the LDVAL register trigger is calculated as number of cycles '1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for timer 2 is enabled by setting PIT*x*_TCTRL2[TIE], the chain mode is activated by setting CHN, and the timer is started by writing a '1' to PIT*x*_TCTRL2[TEN]. In PIT*x*_TCTRL1, the TEN bit needs to be set, and CHN and TIE bits are cleared.

The following example code matches the described setup:

```
                  // turn on PIT
                PIT_CTRL = 0x00;
                   // Timer 2
PIT_LDVAL2 = 0x00000009; // setup timer 2 for 10 counts
                PIT_TCTRL2 = TIE; // enable timer 2 interrupt
                PIT_TCTRL2 |= CHN; // chain timer 2 to timer 1
                PIT_TCTRL2 |= TEN; // start timer 2
                           // Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup timer 1 for 600 000 000 cycles
                PIPIT_TCTRL1 = TEN; // start timer 1
```

44.7 Example Configuration for the Lifetime Timer

To configure the life timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a '0' to field PITx_MCR[MDIS], then the LDVAL registers need to be set to the maximum value.

The following example code matches the described setup:

```
                        // turn on PIT
                      PIT_CTRL = 0x00;
                        // Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
           PIT_TCTRL1 = 0x0; // disable timer 2 interrupts
            PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
                  PIT_TCTRL1 |= TEN; // start timer 1
                        // Timer 0
 PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0for maximum counting period
                PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime read first LTMR64H then LTMR64L.

$$\text{current\_uptime} = \text{PIT\_LTMR64}H << 64;$$
$$\text{current\_uptime} = \text{current uptime} + \text{PIT\_LTMR64}L$$

Equation 21. Equation for current_uptime

The timer is a down counter.

Chapter 45
GTM Integration (GTMINT) Module

45.1 About this module

45.1.1 Definition

The GTMINT integration unit is a hardware module that:

- Provides reset control, clock control, error-correcting code (ECC) functionality, interrupt and DMA control
- Provides access to GTM-IP's memory-mapped registers
- Prevents destructive read accesses into the GTM-IP registers from GTMDI

45.1.2 Features

GTMINT has the following features:

- [Covers: Saf1306][Covers: Saf1317] ECC encoder/decoder for each of 17 RAM interfaces
    - [Covers: Saf1776][Covers: Saf1311] Parity calculation using the whole word and the target memory address value
    - [Covers: Saf1309]Detection and correction of single-bit error on the read data value
    - Detection of single-bit error on address field but no need for correction
    - Double-bit error detection on data or address field without error correction
    - [Covers: Saf1771] ECC report generation for each RAM containing:
        - Address of the error in the register map

- Noncorrectable error indication (not correctable data + address ECC detected errors, or error with the RAM latched address)
- Correctable error indication (correctable data error)

- [Covers: Saf1352]RAM address error detection by comparing latched address from RAM and internal targeted RAM address
- [Covers: ADD21.004]Generation of 225 interrupt requests and or DMA requests
- [Covers: ADD21.007] Debugger access to read all GTM-IP registers through the peripheral bus even when the timer is running. The debugger can read the registers without destroying or changing their contents
- [Covers: ADD21.009] Debugger access to read all GTM-IP RAM addresses read-/writable from the peripheral bus when GTM-IP is running or halted

45.1.3 Modes of operation

GTMINT supports the following modes of operation:

Mode	Description
Normal	The clock signals to GTMINT, GTM-IP, and GTMDI are enabled. The GTM subsystem is functional, processing input timing signals and generating timed outputs.
Stop	The clock signals to GMTINT and GTM-IP are disabled. The only field you can write to is GTMMCR[MDIS].

NOTE

After reset, the mode of GTMINT is defined by the reset value of GTMMCR[MDIS].

45.1.4 Entering Normal mode

To enter Normal mode:

1. Write 0 to GTMMCR[MDIS].

   GTMMCR[STPS] is automatically cleared.

45.1.5 Entering Stop mode

To enter Stop mode:

1. Write 1 to GTMMCR[MDIS].
2. Read GTMMCR[STPS] to confirm the stop request was granted because GTMINT only requests to stop the clocks when no bus accesses to GTM-IP are pending.

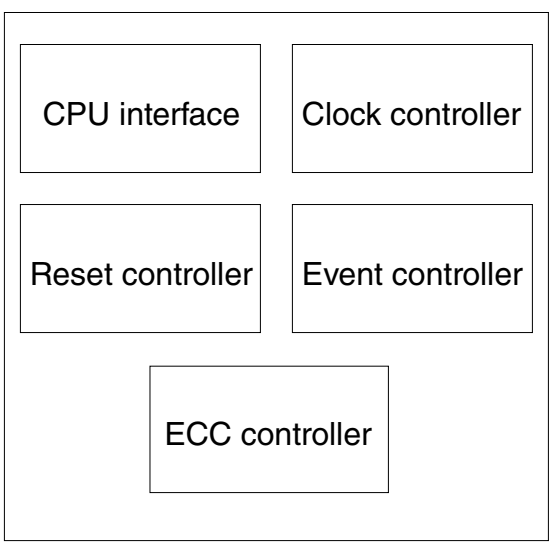
NOTE

The GTM subsystem also enters Stop mode when the chip enters Stop mode. The minimum time between Stop mode requests from the chip is 20 bus clock cycles to ensure that the previous stop mode request acknowledgement was completed by the GTM subsystem.

45.2 GTMINT

45.2.1 GTMINT block diagram

This diagram shows the components of GTMINT:



45.2.2 GTMINT components

This table describes the components of GTMINT:

Component	Function
Clock controller	Manages the clock signals. Enables and disables GTMINT and GTM-IP by gating the peripheral clocks. Clock signals come from the chips clocking module.
Event controller	Routes the GTM-IP interrupt signals to the chip's interrupt controller and DMA controller modules. Controls the interrupt flag clearing. Receives and processes the DMA_DONE_*dmaSource* signal. Does not enable or configure interrupts or DMA events.
CPU interface	Connects GTM-IP's address and data bus to the chip's address and data bus. Signals are connected externally to the chip peripheral bridge and the GTM-IP.
Reset controller	Gives GTMINT maskable software reset control for the GTM-IP to the chip interface.
ECC controller	Generates the parity bits for RAM writes and detects errors in the data read from the RAMs. Signals are connected externally to the GTM's tightly coupled RAMs and the chip's memory manager module.

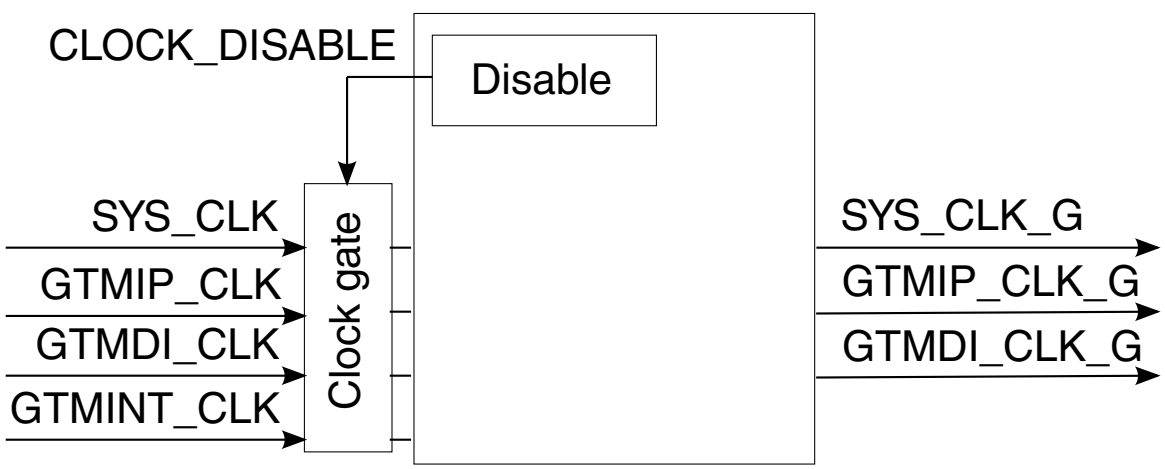
NOTE

GTM-IP is the generic name given to the GTM module used on the chip (for example, GTM104 or GTM103).

45.3 Clock controller

45.3.1 Clock controller block diagram

This diagram shows the clock controller (the clock gate is not part of the clock controller):



45.3.2 Clock controller components

This table describes the components of the clock controller:

Component	Function
Disable	Opens and closes the clock gate (located in the GTMMCR register), which places GTMINT and GTM-IP in Normal mode and Stop mode.
Clock gate	Gates clocks outside the clock controller but is controlled by Disable (GTMMCR[MDIS]).

45.3.3 Clock controller signals

This table describes the clock controller signals:

Signal	I/O	Max. frequency	Function
Clock control			
CLOCK_DISABLE	O	—	Disables the clock gate. This signal is asserted out of reset, placing GTMINT and GTM-IP in Stop mode.
System clock			
SYS_CLK/SYS_CLK_G	I/O	100 MHz	Provides the clock for the address and data bus for memory-mapped register access through the peripheral bridge and AEIMUX. This clock is unaffected by Halt mode. This clock is stopped when the GTM subsystem is in Stop mode.
Peripheral clocks			
GTMIP_CLK/GTMIP_CLK_G	I/O	80 MHz	Provides the clock for GTM-IP timer operation. This clock is stopped when the GTM subsystem is in Stop mode and Halt mode.
GTMDI_CLK/GTMDI_CLK_G	I/O	80 MHz	Provides the clock for GTMDI. This clock is unaffected by Halt mode. This clock is stopped when the GTM subsystem is in Stop mode.
GTMINT_CLK	I	80 MHz	Provides the clock for GTMINT, the GTM-IP registers and the RAM. This clock is unaffected by Halt mode. This clock is stopped when the GTM subsystem is in Stop mode.

45.3.4 Important guidelines for selecting clocks

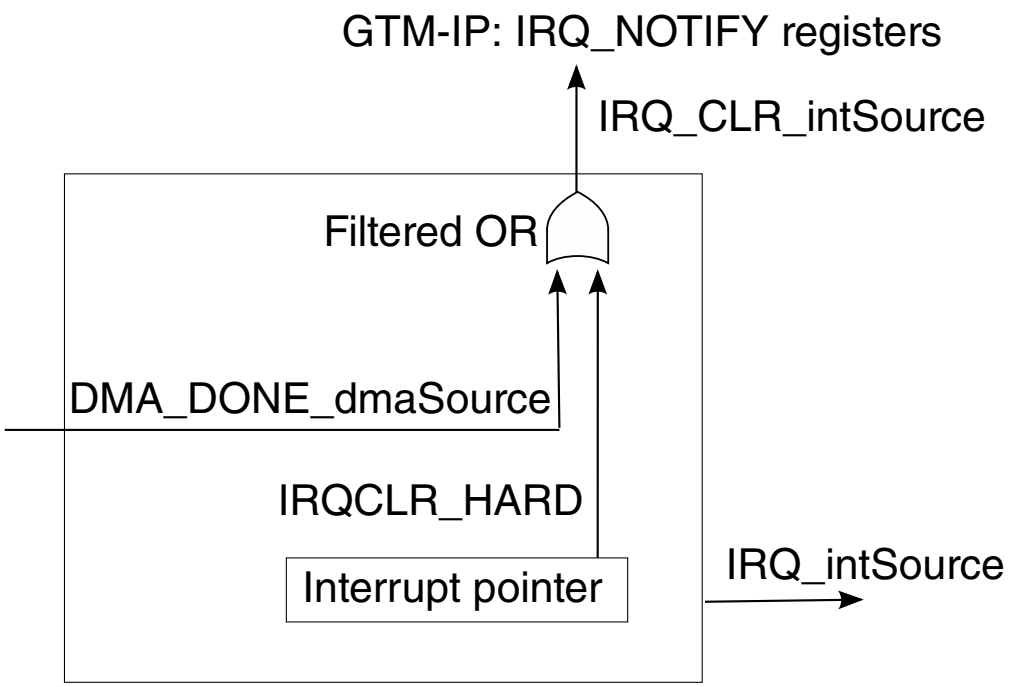
Follow these guidelines for SYS_CLK and the peripheral clocks:

- If your application requires GTM-IP to run in Synchronous mode, ensure that SYS_CLK and the peripheral clocks have the same frequency and phase.
- If your application requires GTM-IP to run in Asynchronous mode, ensure that the frequency of SYS_CLK is greater than or equal to the peripheral clock; the phases of these clocks do not have to be the same.

45.4 Event controller

45.4.1 Event controller block diagram

This diagram shows the event controller:



45.4.2 Event controller components

This table describes the components of the event controller:

Component	Function
Interrupt pointer	Asserts IRQCLR_HARD based on the value of the pointer (GTMINTCLR[INTPTR]) provided by the core.
Filtered OR	Combines the DMA_DONE signals (from the DMA controller) and IRQCLR_HARD and prevents them from introducing unintended state changes (caused by clock domain differences) on the IRQ_CLR signals.

45.4.3 Event controller signals

This table describes the event controller signals:

Signal	I/O	Function
IRQCLR_HARD	I	Initiates the interrupt clear request. Initiated by interrupt-pointer write accesses.
IRQ_CLR_*intSource*[1]	O	Clears the GTM-IP submodule interrupt request.
IRQ_*intSource*[1]	O	Raises an interrupt request to the core's interrupt controller or DMA controller.
DMA_DONE_*dmaSource*[2]	I	Initiates the interrupt clear request. Initiated by the DMA controller when the DMA transfer is completed.

1. *intSource* represents one of the following GTM subsystem interrupt sources: ERR, AEI, ARU, BRC, SPE, CMP, PSM, DPLL, MCS, TIM, TOM, ATOM.
2. *dmaSource* represents one of the following GTM subsystem DMA sources: ERR, AEI, ARU, BRC, SPE, CMP, PSM, DPLL, MCS, TIM, TOM, ATOM.

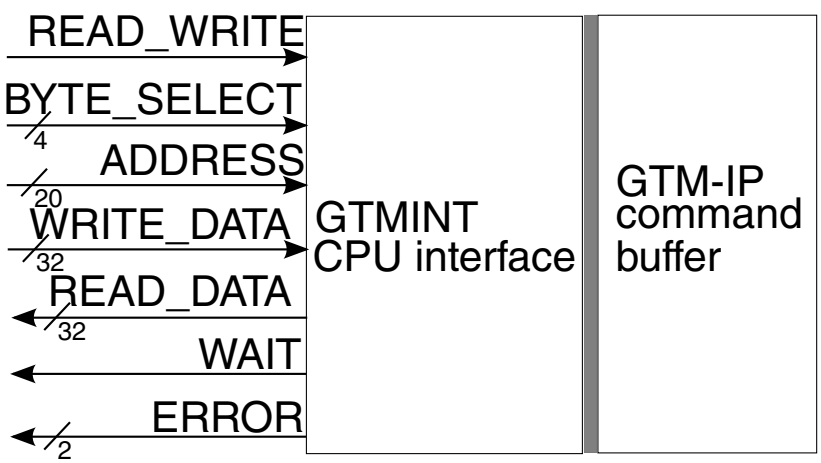
NOTE

The Interrupt Concentrator Module (ICM) bundles interrupt sources inside GTM-IP. This bundling also applies to DMA sources.

45.5 CPU interface

45.5.1 CPU interface block diagram

This diagram shows the CPU interface (the command buffer is not part of the CPU interface):



45.5.2 CPU interface signals

This table describes the CPU interface signals:

Signal	I/O	Function
READ_WRITE	I	Indicates whether the current command from the CPU to GTMI-IP is a data write or data read request.
BYTE_SELECT	I	Selects which byte (or bytes) of data within the 32-bit word is to be read from GTM-IP or GTMINT by the CPU. All bytes are selected when reading a GTM_IP register When the CPU requests a read from a GTMINT register any combination of bytes can be read.
ADDRESS	I	Specifies the 20-bit address of the word that the CPU wants to access.
WRITE_DATA	I	Carries write data from the CPU to GTM-IP or GTMINT.
READ_DATA	O	Carries read data from GTM-IP or GTMINT to the CPU.
WAIT	O	Indicates that the GTM-IP command buffer is full and GTM-IP is fulfilling a CPU read or write command. The peripheral bridge port uses this indication to delay further CPU requests until there is room in the command buffer.
ERROR	O	Indicates the current command has resulted in an error. See your GTM-IP reference manual—for example, *GTM104 Reference Manual*, document GTM104RM—for a description of the potential causes, such as Illegal Byte Addressing, Illegal Address Access, or Unsupported Address.

NOTE

A *request* is a command that may or may not be executed.

A *transfer* is the result of a successfully executed command.

45.5.3 Important guidelines for configuring the CPU interface

This table describes the typical number of wait cycles for accesses through the CPU interface:

Read accesses to	Wait cycles[1]
GTMINT registers	Zero. Commands that access the GTMINT register address range (C0h to FCh) are not added to the command buffer and executed immediately.[2]
GTM-IP registers	Up to 12

Table continues on the next page...

Read accesses to	Wait cycles[1]
MCS RAMs	Up to 12
DPLL RAMs	Up to 13

1. These approximated values are obtained for a core clock frequency equal to 100 MHz and GTM-IP clock frequency equal to 80 MHz. They can change if the ratio of these frequencies is changed.
2. This is also valid for write commands.

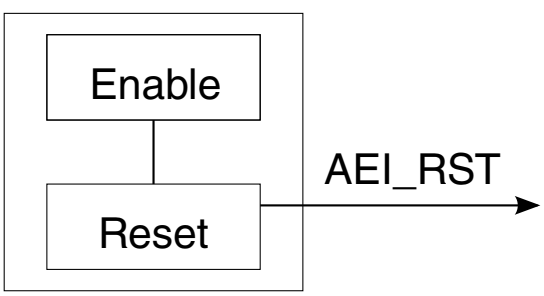
Write operations from chip cores that are executed through GTM-IP's command buffer can be executed differently depending on the configuration of GTM-IP (BRIDGE_MODE[MSK_WR_RSP]).

BRIDGE_MODE[MSK_WR_RSP]	Meaning
1	Does not return error indication. The error is indicated by AEI_IRQ.
	The command buffer is used to receive several write commands until the buffer is full. While the buffer is not full, the slave bus indicates no wait requests. When the buffer is full, a write command generates wait states to the bus master.
0 (default)	Each write operation is executed and generates the error indication and wait cycles until the operation has been executed. The error is also indicated by AEI_IRQ.
	The multiple entries of the command buffer are not used because the bus does not send a new request until the previous one has finished.

45.6 AEI reset

45.6.1 Reset controller block diagram

This diagram shows the reset controller:



45.6.2 Reset controller components

This table describes the components of the reset controller:

Component	Function
Enable	Enables soft reset signaling (GTMMCR[AEISREN]) to the Automotive Electronics Interface (AEI).
Reset	Asserts a soft reset (GTMAEICR[AEIRST]).

45.6.3 Reset controller signals

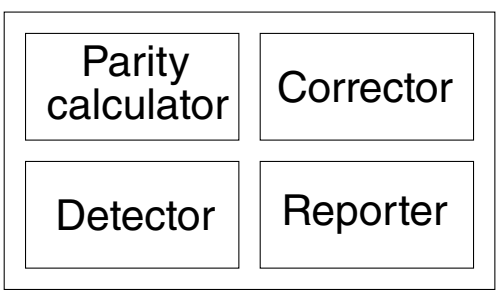
This table describes the reset controller signals:

Signal	I/O	Function
AEI_RST	O	Initiates the Automotive Electronics Interface (AEI) reset sequence.

45.7 ECC controller

45.7.1 ECC controller block diagram

This diagram shows the error-correcting code (ECC) controller:



45.7.2 ECC controller components

This table describes the components of the error-correcting code (ECC) controller:

Component	Used for memory	Function
Parity calculator	Writes	Calculates parity bits and appends them to the original data in the specified memory address. [Covers: Saf1776][Covers: Saf1311] Data and address are considered in the parity-bit calculation. [Covers: Saf1309] This ECC is based on the Hsiao code[1] with 7 parity-bits for each data word.
Detector	Reads	Detects memory errors on reads.

Table continues on the next page...

Component	Used for memory	Function
Corrector	Reads	Uses the ECC data to correct single-bit errors when signaled by the Detector on reads.
Reporter	Reads	Generates and reports the error information when an error is detected.

1. Hsiao, M.Y;.A Class of Optimal Minimum Odd-weight-column SEC-DED Codes; *IBM Journal of Research and Development* [July 1970]

45.7.3 Automatic GTM RAM initialization

Each location in each memory served by the error-correcting code (ECC) controller must be initialized because parity bits need to be stored in each memory address before any read operation. GTM-IP does this automatically for all memories, except for first-in first-out (FIFO) RAM that must be written before being read.

This table presents a summary of GTM-IP memories and the ECCs for single-bit error correction and double-bit error detection.

Memory space	Number of words	Address width (bits)	Data width (bits)	ECC width (bits)	MEM data in/out (data + ECC) (bits)	Matrix Size (number of syndromes)
FIFOx RAM	**1024**	**10**	29	7	**36**	46
DPLL RAM1A	**1286**	**7**	24	7	**31**	38
DPLL RAM1BC	**384**	**9**	24	7	**31**	40
DPLL RAM2	**4096**	**11**	24	7	**31**	42
MCSx RAM0	**1024**	**10**	32	7	**39**	49
MCSx RAM1	**512**	**9**	32	7	**39**	48

45.7.4 ECC syndromes

This table shows the definition of the syndrome field (SYND[6:0]) for each bit position.

NOTE

The error-correcting code (ECC) data-vector width can be different for each memory, so the detector considers only the necessary positions for each one, not all the table positions.

SYND[6:0] / element value	Error result on	Bit number		SYND[6:0] / element value	Error result on	Bit number
40h	code	6		54h	data	23
20h		5		15h		22
10h		4		16h		21
08h		3		34h		20
04h		2		25h		19
02h		1		26h		18
01h		0		64h		17
79h	data[1]	41		2Ch		16
37h		40		1Ah		15
5Bh		39		23h		14
76h		38		2Ah		13
5Dh		37		32h		12
2Fh		36		46h		11
1Fh		35		4Ah		10
70h		34		52h		9
68h		33		62h		8
07h		32		13h		7
49h		31		29h		6
0Dh		30		31h		5
0Eh		29		43h		4
38h		28		45h		3
4Ch		27		19h		2
1Ch		26		51h		1
58h		25		61h		0
0Bh		24		—	—	—
00h	**no error**	—		**any other value**[2]	**noncorrectable error**	—

1. Can be data or address
2. For the FIFOx RAM with 29 data bits and 10 address bits, the valid error positions are from bit 0 to bit 38 only. The syndromes with values for bits 39 to 41 of the table are considered also for noncorrectable errors.

45.7.5 Error report

[Covers: Saf1771] Each RAM read access generates a defined set of information.

Reported information	Description
Error address	The mapped address as defined in the GTM-IP reference manual. This field has meaning only when an error occurs and is flagged by the error signals below.

Table continues on the next page...

Reported information	Description
	It is reported on reads and writes.
Noncorrectable error	An error field indicating that there are detected errors in the read that can't be corrected. The error can occur in the RAM read data bus (parity bits + data) or in the address value that the RAM has received.
Correctable error	An error field indicating that the ECC controller detected and corrected a single-bit error on the RAM read data bus. The parity bits are discarded after the ECC decoder, but this error report includes these bits to obtain the performance of the whole RAM data-storage sector.

The error report remains active for only one RAM clock cycle and should be captured by the external RAM error-monitor module. [Covers: Saf3257] Noncorrectable ECC error indications are provided to GTM-IP, but the reaction to the errors depends on this module itself. For example, if an ECC error occurs while a Multi Channel Sequencer (MCS) channel reads data from a memory page, the channel is disabled and an error bit is asserted. For more information, see your GTM-IP reference manual—for example, *GTM104 Reference Manual*.

45.8 Using GTMINT

45.8.1 Clearing interrupts

45.8.1.1 Deciding whether to use GTMINT or GTM-IP to clear interrupts

This table compares how GTMINT and GTM-IP clear interrupts:

Comparing	GTMINT (See Clearing interrupts via GTMINT.)	GTM-IP (See your GTM-IP reference manual—for example, *GTM104 Reference Manual*, document GTM104RM.)
Interrupt-clearing latency	Low	High[1]
Number of interrupts cleared per request	All interrupts from the same source[2]	Only the requested interrupt[3]
When clearing completes	At the next clock rising edge	Variable, depending on the CPU interface's command buffer depth[4]

1. There is a delay between the command buffer and the actual clearing of the flags inside GTM-IP.
2. Any interrupt from the source that occurs during the interrupt handler, before IRQCLR_HARD is raised, is cleared.

3. Can generate a second interrupt request if the interrupt handler is completed and interrupts are enabled before IRQCLR_SOFT has reached the IRQ_NOTIFY field.
4. It is at the user's discretion whether to wait for the flag to be cleared by polling the *subModule*_IRQ_NOTIFY register or whether to continue with the rest of the program once the clear is initiated.

45.8.1.2 Clearing interrupts via GTMINT

To clear an interrupt source:

1. Write the number for the interrupt source to GTMINTCLR[INTCLR_PTR].

   For example, to clear the IRQ_AEI interrupt, write 2C2h to GTMINTCLR[INTCLR_PTR].

   For a list of interrupts and their corresponding interrupt numbers, see Correspondence between the INTCLR_PTR and the interrupts outputs.

   The event controller asserts the IRQCLR_HARD signal to clear the interrupt.

45.8.1.3 Correspondence between the INTCLR_PTR and the interrupts outputs

INTCLR_PTR[0:9]		Interrupt signal to be cleared
Hexadecimal	Decimal	
000	0	No interrupt is selected
001 to 2C1	1 to 705	Reserved
2C2	706	IRQ_CLR_AEI
2C3 to 2C5	707 to 709	IRQ_CLR_ARU
2C6	710	IRQ_CLR_BRC
2C7	711	IRQ_CLR_CMP
2C8	712	IRQ_CLR_SPE0
2C9	713	IRQ_CLR_SPE1
2CA to 2D1	714 to 721	IRQ_CLR_PSM0
2D2 to 2EC	722 to 748	IRQ_CLR_DPLL
2ED to 2F4	749 to 756	IRQ_CLR_TIM0
2F5 to 2FC	757 to 764	IRQ_CLR_TIM1
2FD to 304	765 to 772	IRQ_CLR_TIM2
305 to 30C	773 to 780	IRQ_CLR_TIM3
30D to 314	781 to 788	IRQ_CLR_MCS0
315 to 31C	789 to 796	IRQ_CLR_MCS1

Table continues on the next page...

INTCLR_PTR[0:9]		Interrupt signal to be cleared
31D to 324	797 to 804	IRQ_CLR_MCS2
325 to 32C	805 to 812	IRQ_CLR_MCS3
32D to 334	813 to 820	IRQ_CLR_TOM0
335 to 33C	821 to 828	IRQ_CLR_TOM1
33D to 344	829 to 836	IRQ_CLR_TOM2
345 to 348	837 to 840	IRQ_CLR_ATOM0
349 to 34C	841 to 844	IRQ_CLR_ATOM1
34D to 350	845 to 848	IRQ_CLR_ATOM2
351 to 354	849 to 852	IRQ_CLR_ATOM3
355 to 358	853 to 856	IRQ_CLR_ATOM4
359	857	IRQ_CLR_SPE2
35A	858	IRQ_CLR_SPE3
35B to 362	859 to 866	IRQ_CLR_PSM1
363 to 36A	867 to 874	IRQ_CLR_TIM4
36B to 372	875 to 882	IRQ_CLR_TIM5
373 to 37A	883 to 890	IRQ_CLR_MCS4
37B to 382	891 to 898	IRQ_CLR_MCS5
383 to 38A	899 to 906	IRQ_CLR_TOM3
38B to 392	907 to 914	IRQ_CLR_TOM4
393 to 396	915 to 918	IRQ_CLR_ATOM5
397 to 39A	919 to 922	IRQ_CLR_ATOM6
39B to 39E	923 to 926	IRQ_CLR_ATOM7
39F to 3A2	927 to 930	IRQ_CLR_ATOM8
3A3	931	IRQ_CLR_ERR
3A4 through 3FF	932 through 1023	Reserved

45.8.2 Resetting the GTM Subsystem

45.8.2.1 Deciding what type of reset to use

This table compares the different types of resets:

Comparing	Submodule channel (See your GTM-IP reference manual—for example, *GTM104 Reference Manual*, document GTM104RM.)	GTM-IP (See your GTM-IP reference manual—for example, *GTM104 Reference Manual*, document GTM104RM.)	AEI (See Resetting the AEI.)	GTM subsystem (See the chip reference manual.)
What is reset	A single submodule channel	GTM-IP	The peripheral bridge interface	The GTM subsystem
What you must reconfigure	The channel	GTM-IP. GTMINT and GTMDI are unaffected.	The GTM-IP GTM_BRIDGE_MODE register	The entire GTM subsystem
Reset protection?	No	Yes. [Covers: ADD13.014]The GTM-IP reset (GTM_RST[RST]) is write protected by GTM_CTRL[RF_PROT].	Yes. The AEI reset (GTMAEICR[AEIRST]) is write protected by GTMMCR[AEISREN].	No
System dependencies?	No	No	Yes. Use this type of reset only when the buffer is empty or in an error state.	No

45.8.2.2 Resetting the AEI

NOTE

Reset the GTM-IP bridge reset (GTM_BRIDGE_MODE[BRG_RST]) before resetting the AEI.

To reset the Automotive Electronics Interface (AEI):

1. Write 1 to GTMMCR[AEISREN].

   The reset controller enables the AEI reset functionality by un-protecting the GTMAEICR[AEIRST] field.
2. Write 1 to GTMAEICR[AEIRST].

   The reset controller asserts the AEI_RST signal in the next clock cycle and takes one bus clock cycle to complete.

   The GTMINT clears the AEI buffer, resets the configuration registers, aborts slave bus accesses (which means that access commands in the buffer memory are lost).

45.9 Memory map and registers

45.9.1 Memory map

This section contains the description of additional registers on the integration module. Descriptions of GTM-IP registers, or GTMDI registers are found in their respective documentation.

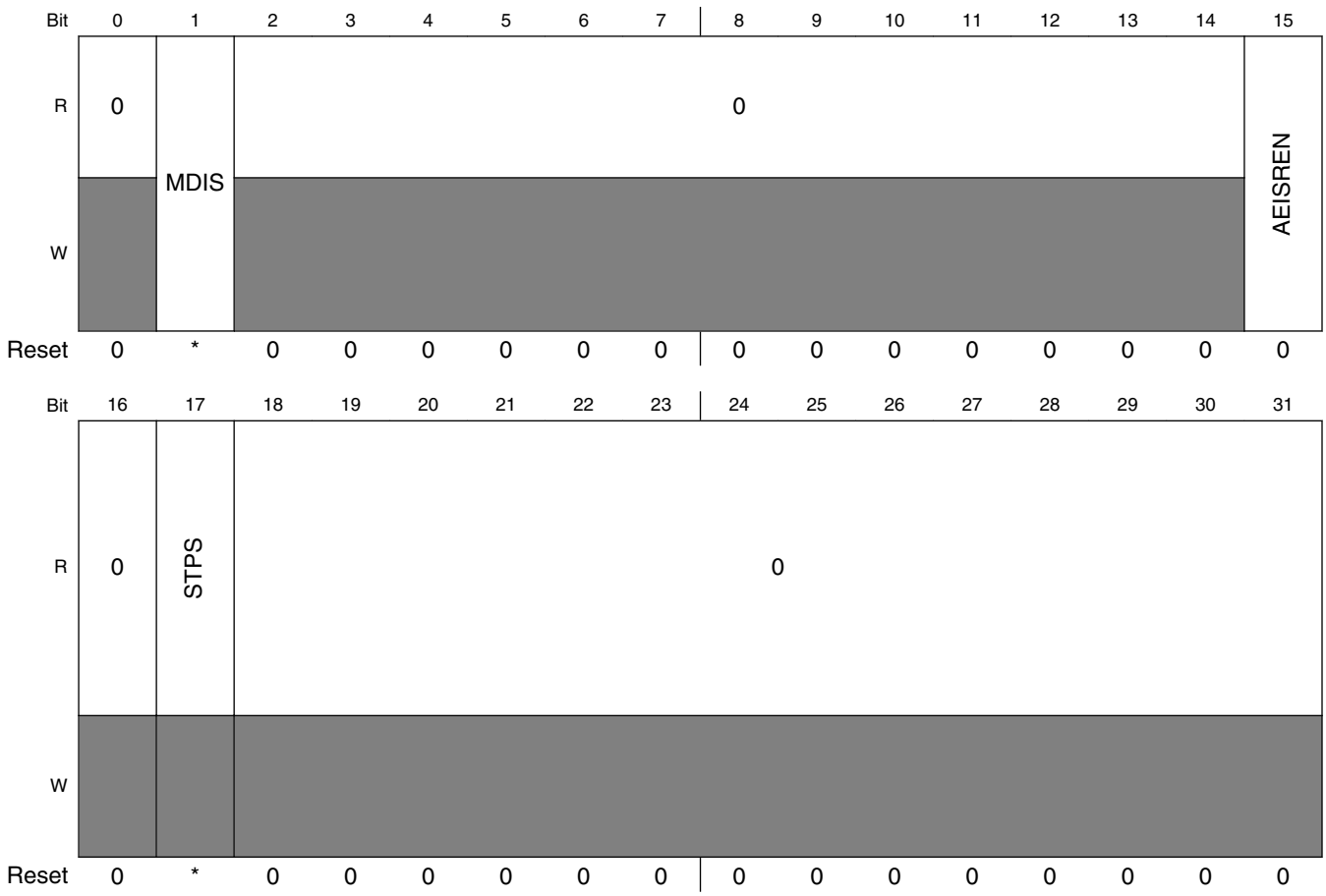
GTMINT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C0	GTM Module Configuration Register (GTMINT_GTMMCR)	32	R/W	See section	45.9.1.1/ 1728
C8	GTM Interrupts Clear Register (GTMINT_GTMINTCLR)	32	W	0000_0000h	45.9.1.2/ 1729
CC	GTM AEI Control Register (GTMINT_GTMAEICR)	32	W	0000_0000h	45.9.1.3/ 1730

45.9.1.1 GTM Module Configuration Register (GTMINT_GTMMCR)

GTMMCR contains the control bits to configure the general operation of GTMINT and GTM-IP.

Address: 0h base + C0h offset = C0h



* Notes:
- STPS field: Reset value is chip dependent.
- MDIS field: Reset value is chip dependent.

GTMINT_GTMMCR field descriptions

Field	Description
0 Reserved	This read-only field is reserved and always has the value 0.
1 MDIS	[Covers: Saf1461] MDIS Module Disable.

Table continues on the next page...

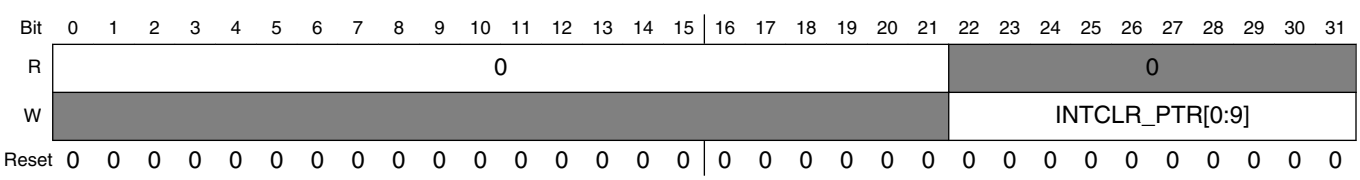
GTMINT_GTMMCR field descriptions (continued)

Field	Description
	MDIS puts GTMINT in stop mode. Communication through the peripheral bus is ignored in this mode except reads/writes to the GTMMCR register and reads from other integration registers, which are allowed. Refer to Entering Normal mode and Entering Stop mode sections for more details. [Covers: Saf1462] 0 Normal mode 1 Module disable requested
2–14 Reserved	This read-only field is reserved and always has the value 0.
15 AEISREN	Saf1276 - Safety-critical AEISREN AEI interface soft-reset control enable. AEISREN enables the soft-reset of the AEI interface that is controlled by AEISRST bit in GTMAEICR register. Refer to Resetting the AEI section for more details. 0 AEI soft-reset control is disabled 1 AEI soft-reset control is enabled
16 Reserved	This read-only field is reserved and always has the value 0.
17 STPS	[Covers: Saf1461] STPS Stop Mode Status. STPS indicates when GTM-IP is in stop mode. Refer to Entering Stop mode section for more details. 0 Normal Mode 1 GTM subsystem is in Stop Mode
18–31 Reserved	This read-only field is reserved and always has the value 0.

45.9.1.2 GTM Interrupts Clear Register (GTMINT_GTMINTCLR)

GTMINTCLR contains the control bits to clear some specific interrupt request.

Address: 0h base + C8h offset = C8h



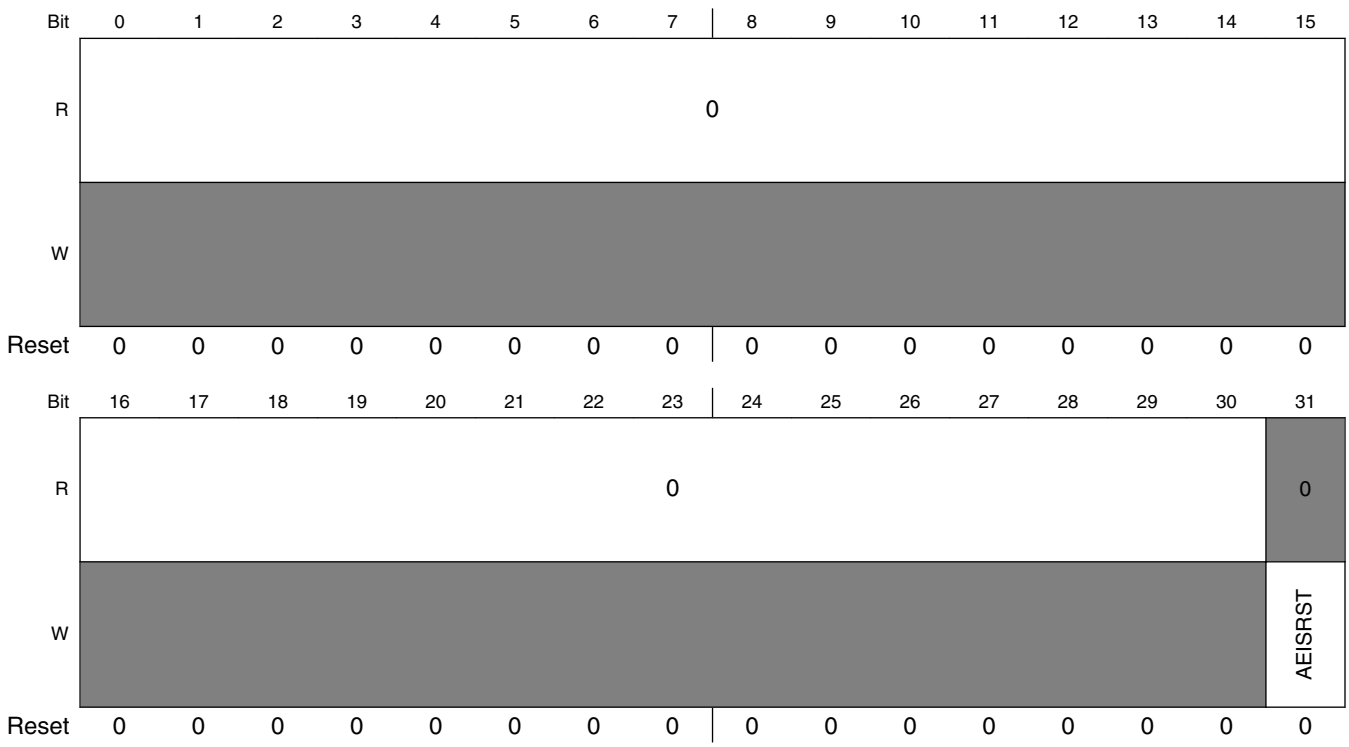
GTMINT_GTMINTCLR field descriptions

Field	Description
0–21 Reserved	This read-only field is reserved and always has the value 0.
22–31 INTCLR_ PTR[0:9]	[Covers: Saf1461] INTCLR_PTR Interrupt Clear Pointer. This field selects which interrupt will be cleared. The correspondence between the number and the interrupt is given in Correspondence between the INTCLR_PTR and the interrupts outputs. Refer to Clearing interrupts via GTMINT for more details.

45.9.1.3 GTM AEI Control Register (GTMINT_GTMAEICR)

GTMAEICR contains the control bits for the AEI interface soft reset.

Address: 0h base + CCh offset = CCh



GTMINT_GTMAEICR field descriptions

Field	Description
0–30 Reserved	This read-only field is reserved and always has the value 0.
31 AEISRST	[Covers: Saf1461] AEISRST

Table continues on the next page...

GTMINT_GTMAEICR field descriptions (continued)

Field	Description
	AEI interface soft-reset control. AEISRST applies a reset signal to GTM-IP AEI interface. The signal is enabled by the AEISREN bit in GTMMCR register. This is a self clear bit and is always read as 0. Refer to Resetting the AEI for more details. 0 No reset 1 Reset applied to GTM-IP AEI interface

Chapter 46
Generic Timer Module (GTM104)

46.1 About this module

GTM104 is a programmable timer that can perform complex timing and I/O management independently of the chip's processor cores.

46.2 For more information

For more information on GTM104, see the *Generic Timer Module 104 (GTM104) Reference Manual*.

Chapter 47
GTM Development Interface (GTMDI)

47.1 About this module

The GTMDI debug module unit is a hardware module that:

- Monitors the software and hardware activities of GTM-IP
- Read the registers without interfering with the logic status, even for FIFOs, by configuring the debug access controls.

47.2 Introduction

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including MCS cores, ARU data trace, Input Channel Filters trace, Output Timer channels, DPLL and SPE submodules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules in order to implement complex debug features. Trace capability is also provided along with a global time-stamp that allows real time trace for several GTM submodule events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and NPC Aurora router on-chip modules. Note that the Nexus standard is only used for the physical interfaces. In order to implement the debug features such as breakpoint and messages required for the GTM debug, the Nexus standard could not be followed.

This figure shows a block diagram of the Generic Timer Module (GTM) Development Interface.

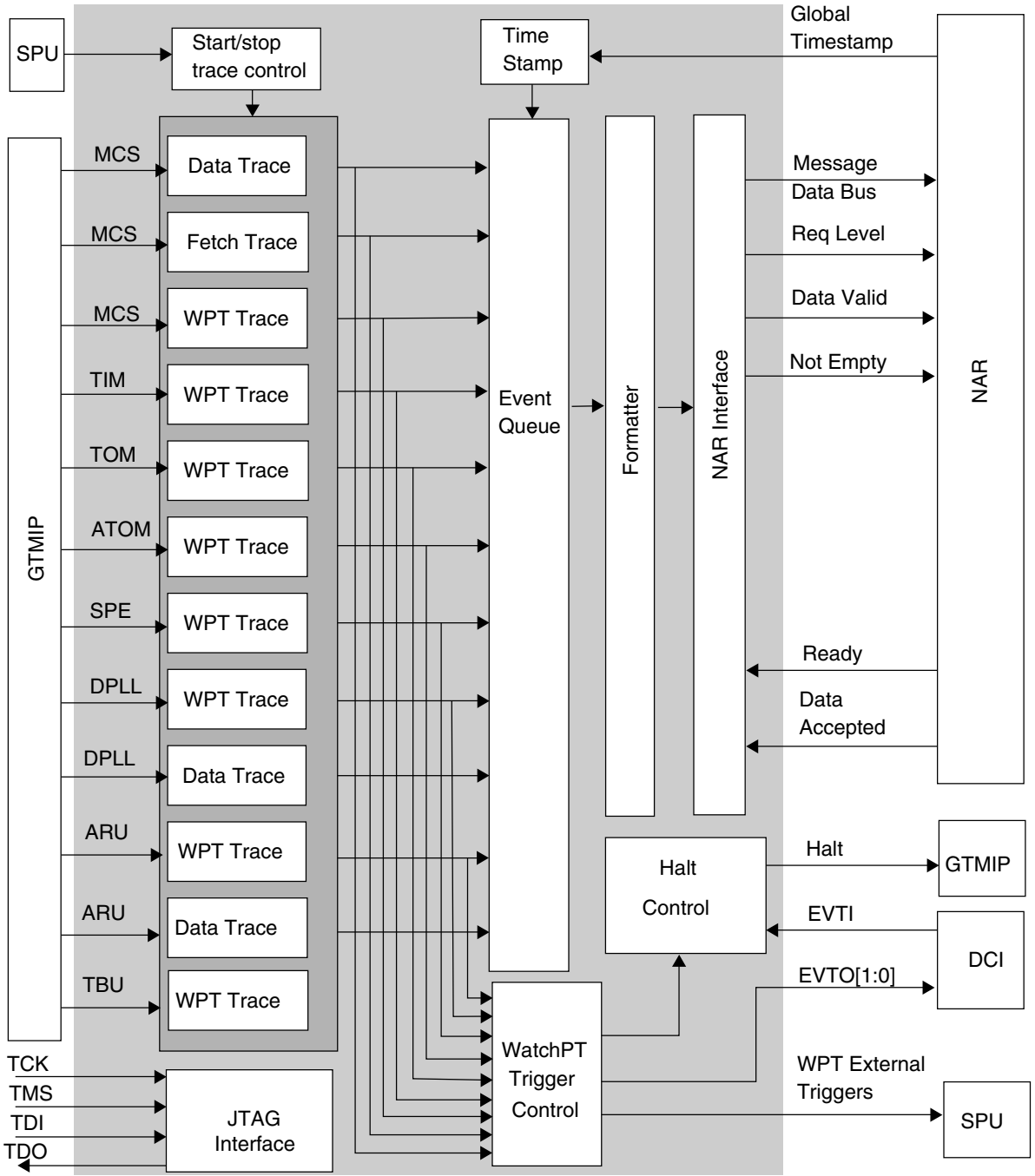


Figure 47-1. GTMDI block diagram

47.3 Overview

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-Channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, input channel filters trace, output timer channels, DPLL and Sensor Pattern Evaluation (SPE) sub-modules. The GTMDI interfaces to the GTM IP

using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM sub-module events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and NAR on-chip modules. The Nexus standard is only used for the physical interfaces. Due to specific requirements for the GTM debug, the Nexus standard is not followed.

The following figure describes the interconnection between the GTMDI and GTM modules. The GTMDI selects which signal or bus to monitor since it does not have the capability to monitor all signals in parallel. The multiplexing architecture is described in the following sections.

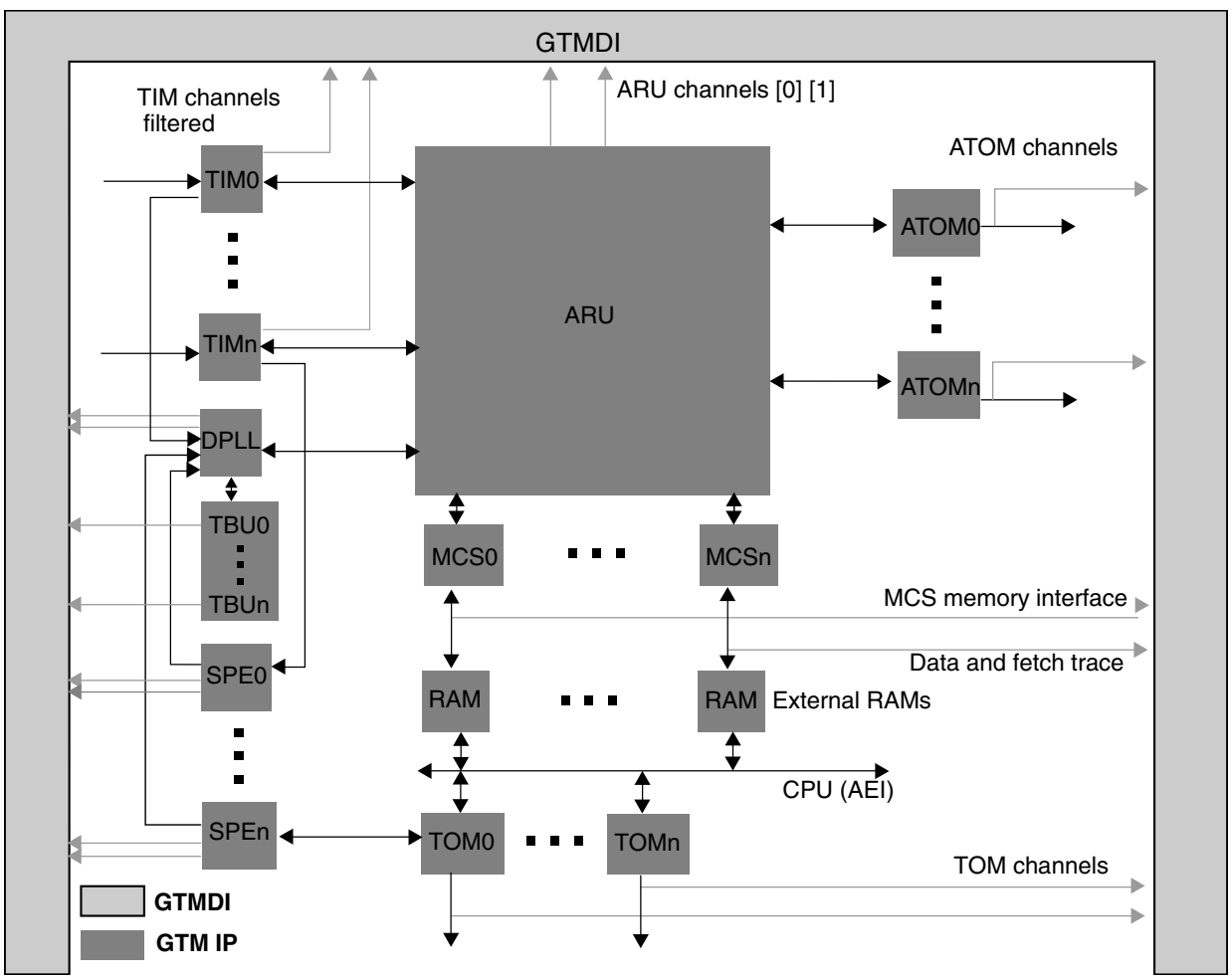


Figure 47-2. GTMDI and GTM interface

Note

Figure 47-2 does not show all of the GTM internal modules. It shows only the modules that interface to the GTMDI debug module.

The number of ATOM, TIM, TOM, MCS and SPE modules in the GTMDI and GTM interface figure is shown in this table.

Table 47-1. GTM Sub-module instances

Module	Number of instances
ATOM	9
TIM	6
TOM	5
MCS	6
SPE	4

The following figure shows the Watchpoint Triggers (WPT) and Watchpoint Messages (WPM). Each GTM sub-module generates two watchpoints which can be routed internally to the GTMDI and generate watchpoint triggers or routed to the message formatter and generate watchpoint messages. The selection between watchpoint triggers and messages is implemented per sub-module. Messages and watchpoint triggers can also be generated at the same time based on the same event within a sub-module, thus a TIM channel transition can generate a watchpoint trigger and a watchpoint message. The watchpoint triggers may be sent to the Sequence Processing Unit (SPU) outside the GTMDI if enabled by appropriate control register bits for each sub-module, such as Timer Input Module (TIM), Timer Output Module (TOM) and ARU-connected Timer Output Module (ATOM). The watchpoint messages, if enabled, are sent to the NAR through the Nexus Client interface.

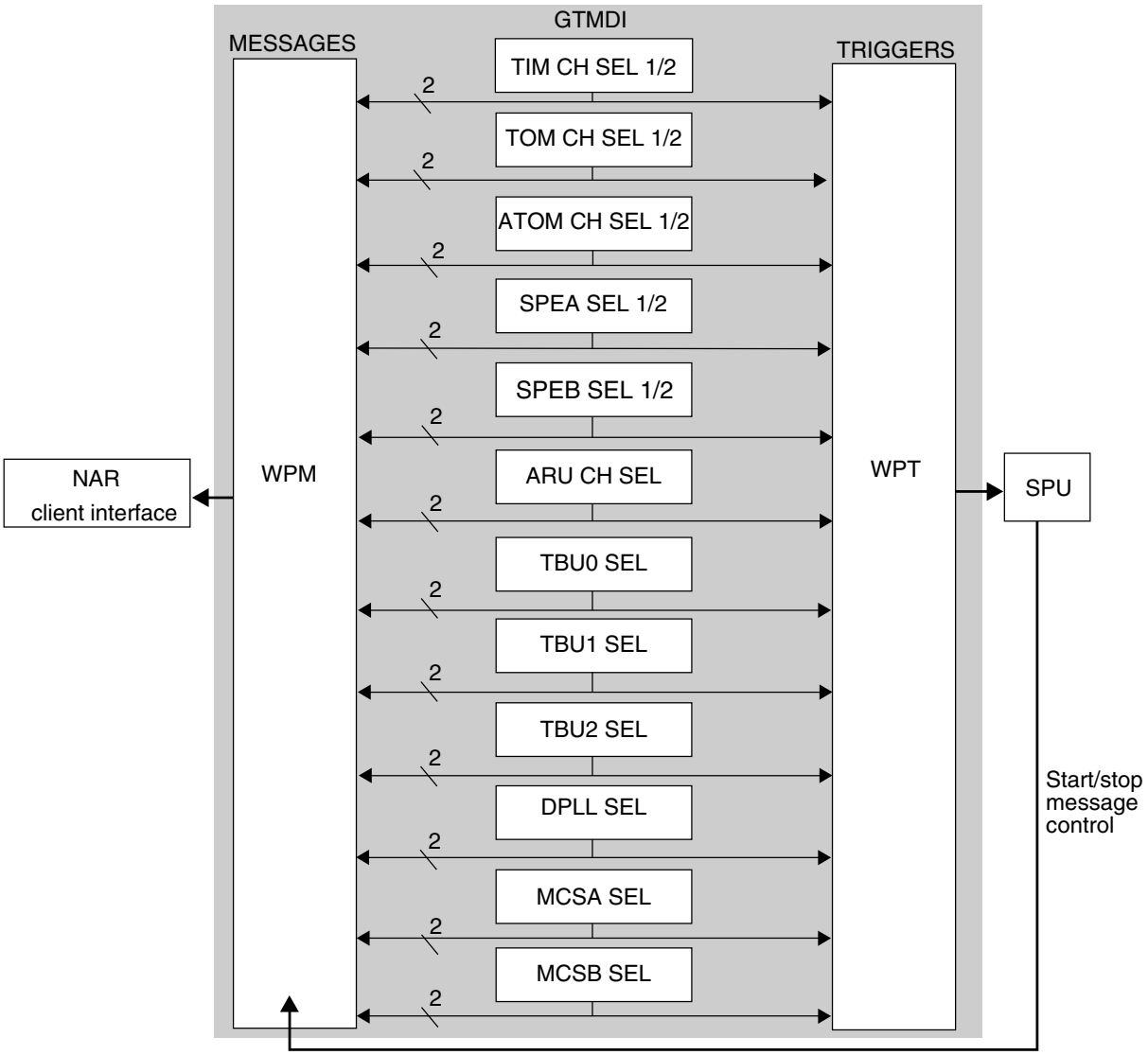


Figure 47-3. Watchpoint triggers and watchpoint messages

The SPU provides external control of the watchpoint messages generated by the GTMDI thus controlling the bandwidth over the Nexus Client interface. This functionality is implemented through Start/Stop control signals from the SPU. These signals are connected to the sub-module register bit that enables the message transmission implementing a dual control over these registers, either by JTAG interface programing or by SPU Start/Stop signals. See Functional description for more details about register control and general architecture.

47.3.1 Features

The GTMDI block implements the following features:

- Full duplex pin interface for medium and high visibility throughput
  - 2 $\overline{\text{EVTO}}$ (Event Out) for GTM real time signal monitoring
  - 1 $\overline{\text{EVTI}}$ (Event In)
  - IEEE 1149.1 (JTAG) Test Access Port (TAP)
    - 4 pins (TDI, TDO, TMS, and TCK)
    - Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
- GTM Development Support
  - Read/Write registers in debug mode from JTAG port
  - Ability to enter debug mode at reset negation or during normal execution

Note

Enter debug out-of-reset is provided through the SoC system level debug control signal. In order to allow this functionality the GDE and SDBE control bits in the GTMDI_DC register should be set.

  - Halt and Watchpoint Configuration
  - MCS Data Trace Messaging (DTM) and Watchpoint Messaging (WPM) allows the development tool to trace reads and writes to selected MCS RAM address ranges
    - Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the Aurora port bandwidth by constraining the number of trace locations.
    - Fetch trace capability controlled by two programmable Watchpoints.
    - Private messages are used to indicate special MCS core cases not covered by public messages. Two MCS cores can be debugged at the same time.

Note

Enabling two MCS cores at the same time for data trace can exceed the capacity of the NAR Client interface throughput thus information can be lost.

    - DPLL Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace reads and writes to selected DPLL RAM addresses and selected data value.
    - ARU Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace selected ARU channels.
    - Watchpoint Messaging (WPM)
- Trigger Control
    - The GTMDI provides selection of trigger signals that generate debug actions at system level such as Halt GTM itself or cores in the same device.
    - Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as Timer Input Module (TIM), Timer Output Module (TOM), ARU-connected Timer Output Module (ATOM), etc.
- GTMDI clock operation
    - The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI operates from the same clock source as the GTM IP module.

47.3.2 Modes of operation

The GTMDI block is placed in reset when the Nexus reset signal is asserted. The JTAG registers controlled by the TCK clock are also put into the reset state if the TAP controller is in the TEST-LOGIC-RESET state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the TEST-LOGIC-RESET state regardless of the initial TAP controller state. Asserting the Nexus reset signal results in asynchronous entry into the reset state not only of the TCK controlled registers but all registers in the GTMDI. The GTMDI is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the TEST-LOGIC-RESET state
- The auxiliary output port pins are negated
- The auxiliary output port enable outputs are negated

- The TDI, TMS, and TCK TAP inputs are ignored (The TMS and TCK signal are ignored only while the Nexus reset signal is asserted.)
- Registers default back to their reset values

Note

The GTMDI is not reset by the system reset signal. It is reset during device Power On reset or by a dedicated Nexus reset thus preserving internal registers state and programming during system reset.

Debug Mode can be controlled by on-chip logic through the system debug signal. The level on this signal defines if the GTM module is in debug mode, meaning that it is in Halt state and accepts write and read accesses to internal registers that do not modify the state of those registers. If asserted, the GTM enters Halt state. If negated, the GTM debug mode is controlled by other sources, internal to the GTMDI. The GTM may enter debug out of reset by asserting the system debug signal while the device is being reset. By detecting this condition, the GTMDI puts the GTM into the Halt state. The GDE and SDBE bits described in GTMDI development control register (GTMDI_DC) must be set during reset allowing the GTM to enter debug state.

47.3.2.1 DATA port

GTMDI messages are sent through the Message Data Bus. The Message Data Bus has a fixed width of 32 bits which defines one BEAT of data. There are three BEATS per NAR clock, thus 96 bits are transmitted in parallel.

47.4 External signal description

The GTMDI pinout provides interface for the transmission of messages and for accessing Nexus registers. The GTMDI pin definition is outlined in the following table.

Table 47-2. GTMDI signal properties

Name	Port	Function	Reset State
$\overline{\text{EVTI}}$	Auxiliary	Event In pin	—
$\overline{\text{EVTO}}$[1:0]	Auxiliary	Event Out pin	[11]
TCK	TAP	Test Clock Input	—
TDI	TAP	Test Data Input	—

Table continues on the next page...

Table 47-2. GTMDI signal properties (continued)

Name	Port	Function	Reset State
TDO	TAP	Test Data Output	0
TMS	TAP	Test Mode Select Input	—
$\overline{\text{TRST}}$	TAP	Test Reset Input	—

Each of the signals listed in this table is described in more detail in the following sections.

47.4.1 Event in ($\overline{\text{EVTI}}$)

Event In ($\overline{\text{EVTI}}$) is used to generate a Halt if enabled by the corresponding control bit. $\overline{\text{EVTI}}$ is an edge-sensitive signal. It is synchronized at the GTMDI input and a falling edge is detected.

47.4.2 Event out $\overline{\text{EVTO}}$[1:0]

Event Out ($\overline{\text{EVTO}}$) are two outputs from GTMDI for development tools with exact timing for a two watchpoints or Halt occurrence. $\overline{\text{EVTO}}$ is asserted for one system clock cycle when Halt or watchpoints are monitored. When internal GTM signals are monitored, such as TIM filtered signals or TOM/ATOM outputs, the $\overline{\text{EVTO}}$ represent the exact signal shape with a certain delay. In this monitoring mode the signal is not inverted through the $\overline{\text{EVTO}}$ output. The real time monitoring mode is intended for oscilloscope type of monitoring thus allowing the user to evaluate the actual shape of the signals. The source selection for the $\overline{\text{EVTO}}$ is centralized in the Development Control register GTMDI_DC.

47.4.3 Test clock input (TCK)

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the JTAG port.

47.4.4 Test data input (TDI)

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

47.4.5 Test data output (TDO)

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is actively driven in the SHIFT-IR and SHIFT-DR states of the JTAG state machine. TDO changes on the falling edge of TCK and is sampled on the rising edge of TCK.

47.4.6 Test Mode Select (TMS)

Test Mode Select (TMS) is an input pin used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCK.

47.5 Register definition

This section provides a detailed description of all GTMDI registers accessible to the development tool. Individual bit-level descriptions and reset states of each register are included. The following table shows the GTMDI registers and index values. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 47-3. GTMDI registers

Register	Index	Read/write
Device Identity (DID)	0	R
Reserved	1–4	—
GTMDI Development Control (GTMDI_DC)	5	R/W
GTMDI Development Status (GTMDI_DS)	6	R
Reserved	7–10	—
TIM Watchpoint Control 1 (GTMDI_TIM_WPC1)	11	R/W
TIM Watchpoint Control 2 (GTMDI_TIM_WPC2)	12	R/W
Reserved	13–16	—
TOM Watchpoint Control 1 (GTMDI_TOM_WPC1)	17	R/W
TOM Watchpoint Control 2 (GTMDI_TOM_WPC2)	18	R/W
Reserved	19–22	—
ATOM Watchpoint Control 1 (GTMDI_ATOM_WPC1)	23	R/W
ATOM Watchpoint Control 2 (GTMDI_ATOM_WPC2)	24	R/W
Reserved	25–28	—
SPEA Watchpoint Control 1 (GTMDI_SPEA_WPC1)	29	R/W
SPEA Watchpoint Control 2 (GTMDI_SPEA_WPC2)	30	R/W
Reserved	31–34	—

Table continues on the next page...

Table 47-3. GTMDI registers (continued)

Register	Index	Read/write
SPEB Watchpoint Control 1 (GTMDI_SPEB_WPC1)	35	R/W
SPEB Watchpoint Control 2 (GTMDI_SPEB_WPC2)	36	R/W
Reserved	37–40	—
DPLL Development Control (GTMDI_DPLL_WPC1)	41	R/W
DPLL Watchpoint Control 1 (GTMDI_DPLL_WPC2)	42	R/W
DPLL Watchpoint Control 2 (GTMDI_DPLL_WPC3)	43	R/W
DPLL Watchpoint Control 3 (GTMDI_DPLL_WPC4)	44	R/W
DPLL Watchpoint Control 4 (GTMDI_DPLL_WPC5)	45	R/W
DPLL Data Trace Control (GTMDI_DPLL_DTC)	46	R/W
Reserved	47–49	—
ARU Watchpoint Control (GTMDI_ARU_WPC1)	50	R/W
ARU Watchpoint Control (GTMDI_ARU_WPC2)	51	R/W
ARU DATA0H (GTMDI_ARU_DATA0H)	52	R/W
ARU DATA0L (GTMDI_ARU_DATA0L)	53	R/W
ARU DATA1H (GTMDI_ARU_DATA1H)	54	R/W
ARU DATA1L (GTMDI_ARU_DATA1L)	55	R/W
ARU Data Trace Control (GTMDI_ARU_DTC)	56	R/W
Reserved	57–59	—
MCSA Development Control (GTMDI_MCSA_DC)	60	R/W
Reserved	61	—
MCSA Watchpoint Control (GTMDI_MCSA_WPC)	62	R/W
MCSA Program Fetch Trace Control (GTMDI_MCSA_PTC)	63	R/W
MCSA Data Trace Control (GTMDI_MCSA_DTC)	64	R/W
MCSA Watchpoint Address 1 (GTMDI_MCSA_WPA1)	65	R/W
MCSA Watchpoint Address 2 (GTMDI_MCSA_WPA2)	66	R/W
MCSA Watchpoint Data 1 (GTMDI_MCSA_WPD1)	67	R/W
MCSA Watchpoint Data 2 (GTMDI_MCSA_WPD2)	68	R/W
MCSA Program Trace Channel Enable (GTMDI_MCSA_CE)	69	R/W
Reserved	70	—
MCSA Data Trace Address Range 1 (GTMDI_MCSA_DTAR1)	71	R/W
MCSA Data Trace Address Range 2 (GTMDI_MCSA_DTAR2)	72	R/W
Reserved	73–76	—
MCSB Development Control (GTMDI_MCSB_DC)	77	R/W
Reserved	78	—
MCSB Watchpoint Control (GTMDI_MCSB_WPC)	79	R/W
MCSB Program Fetch Control (GTMDI_MCSB_PTC)	80	R/W
MCSB Data Trace Control (GTMDI_MCSB_DTC)	81	R/W
MCSB Watchpoint Address 1 (GTMDI_MCSB_WPA1)	82	R/W
MCSB Watchpoint Address 2 (GTMDI_MCSB_WPA2)	83	R/W

Table continues on the next page...

Table 47-3. GTMDI registers (continued)

Register	Index	Read/write
MCSB Watchpoint Data 1 (GTMDI_MCSB_WPD1)	84	R/W
MCSB Watchpoint Data 2 (GTMDI_MCSB_WPD2)	85	R/W
MCSB Program Trace Channel Enable (GTMDI_MCSB_CE)	86	R/W
Reserved	87	—
MCSB Data Trace Address Range 1 (GTMDI_MCSB_DTAR1)	88	R/W
MCSB Data Trace Address Range 2 (GTMDI_MCSB_DTAR2)	89	R/W
Reserved	90–93	—
TBU0 Watchpoint Control 1 (GTMDI_TBU0_WPC1)	94	R/W
TBU0 Watchpoint Control 2 (GTMDI_TBU0_WPC2)	95	R/W
TBU0 DATA (GTMDI_TBU0_DATA)	96	R/W
Reserved	97–100	—
TBU1 Watchpoint Control 1 (GTMDI_TBU1_WPC1)	101	R/W
TBU1 Watchpoint Control 2 (GTMDI_TBU1_WPC2)	102	R/W
TBU1 DATA (GTMDI_TBU1_DATA)	103	R/W
Reserved	104–107	—
TBU2 Watchpoint Control 1 (GTMDI_TBU2_WPC1)	108	R/W
TBU2 Watchpoint Control 2 (GTMDI_TBU2_WPC2)	109	R/W
TBU2 DATA (GTMDI_TBU2_DATA)	110	R/W

47.5.1 Register descriptions

This section consists of GTMDI register descriptions. The registers are shown in Table 47-3 . The MCS cores selected for debugging are named MCSA and MCSB thus two MCS cores can be debugged at the same time. See registers GTMDI_MCSA_DC and GTMDI_MCSB_DC which specify the real number of the MCS associated with MCSA and MCSB.

47.5.1.1 Device identity register (DID)

The DID register contains the unique identification information as shown in this table.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
RESET	DID_PRN				DID_DC						DID_PIN					

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code											1
W																
RESET	DID_PIN (contd)				DID_MIC											1

This table describes the device identification register functions.

Table 47-4. DID field descriptions

Field	Description
31–28 PRN	Part Revision Number. Contains the revision number of the part. Value is 0x1.
27–22 DC	Design Center. Indicates the design center. Value is 0x2b.
21–12 PIN	Part Identification Number. Contains the part number of the device. Value is 0x3E4.
11–1 MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
0 IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

47.5.1.2 GTMDI development control register (GTMDI_DC)

The GTMDI_DC register controls various trace and debug features of the GTM modules as described in this table.

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GDE	0	0	0	EOS0				0	0	0	0	EOS1			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	EIDR	0	DBE	DBR	SDBE	0	TSFT
W								CHR								
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DC register functions.

Table 47-5. GTMDI_DC field descriptions

Field	Description
31 GDE	Global Debug Enable. This control bit disables all debug features of the GTMDI module including ability to put GTM IP in Halt state, modify EVTO output, send messages to the DATA bus and set triggers to the SPU module. Before changing the GTMDI configuration it is recommended to clear GDE bit to prevent spurious behavior of the module outputs. See Figure 47-4 for details about debug enable logic. 0 GTMDI Global debug is disabled 1 GTMDI Global debug is enabled
27–24 EOS0	$\overline{\text{EVTO0}}$ Output Selection. This signal is dedicated to sel 1 which is the selection 1 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output EVTO0 remains at one 0001 Halt Indication, If $\overline{\text{EVTO0}}$ is zero indicates that GTM is halted 0010 TIM, Map EVTO0 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO0 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO0 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO0 from SPEA, controlled by TSS1 described in SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1) 0110 SPEB, Map EVTO0 from SPEB, controlled by TSS1 described in SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1) 0111 ARU CH0, Map EVTO0 from ARU CH0l 1000 DPLL, Map EVTO0 from DPLL according to TSEL1 in DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1) 1001 MCSA, Map EVTO0 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO0 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved
19–16 EOS1	$\overline{\text{EVTO1}}$ Output Selection. This signal is dedicated to sel 2 which is the selection 2 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output EVTO1 remains at one 0001 Halt Indication, If $\overline{\text{EVTO1}}$ is zero indicates that GTM is halted 0010 TIM, Map EVTO1 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO1 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO1 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO1 from SPEA, controlled by TSS2 described in SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1) 0110 SPEB, Map EVTO1 from SPEB, controlled by TSS2 described in SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1)

Table continues on the next page...

Table 47-5. GTMDI_DC field descriptions (continued)

Field	Description
	0111 ARU CH1, Map EVTO1 from ARU CH1 1000 DPLL, Map EVTO1 from DPLL according to TSEL2A/B in DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1) 1001 MCSA, Map EVTO1 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO1 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved
8 CHR	Clear Halt Request. Writing this bit to one clears halt requests., which means GTM resumes normal operation out of halt state. The status bits of the GTMDI_DS register are cleared. This bit is self-cleared thus read always as zero. 0 No action 1 Clear halt requests
6 EIDR	EVTI Debug Request enable. The EIDR bit enables EVTI input to request the GTM to enter Halt state. After the GTM enters Halt state due to a pulse in EVTI signal, the EIDR bit state is not meaningful, thus it may be cleared and the GTM continues in Halt state. See Figure 47-4 for details about debug enable logic. 0 Disable EVTI to control GTM to enter Halt state 1 Enable EVTI to control GTM to enter Halt state
4 DBE	Debug Enable. DBE enables debug mode thus allowing the GTM to enter debug mode which means halt regular operation. The debug events controlled by this register are the ones generated internally to the GTMDI, excluding ipg_debug and EVTI which have their dedicated enable control. See Figure 47-4 for details about debug enable logic. 0 Debug mode disabled 1 Debug mode enabled
3 DBR	Debug Request. Mechanism to allow development tool to request modules to enter debug mode directly. See Figure 47-4 for details about debug enable logic. 0 Does not modify GTM IP debug state 1 Request GTM IP to enter debug mode
2 SDBE	System Debug Enable. The SDBE bit enables the GTM to enter halt state if system level debug enable signal is asserted. This bit allows the debug controller to select specific sub-groups of cores or modules to halt, while others continue to operate. GDE bit must also be set to allow GTM to halt. See Figure 47-4 for details about debug enable logic. 0 GTM does not enter halt state due to system debug enable signal assertion 1 GTM is enabled to enter halt state due to system debug enable signal assertion
0 TSFT	Timestamp Force Transmission. Forces the transmission of the Timestamp value along with the Nexus message data for all messages transmitted by the GTMDI Client module. When Timestamp inclusion is not forced into the client message, its inclusion depends upon the NAR settings. 0 Does not force transmission of Timestamp value 1 Forces transmission of Timestamp along with Nexus message

47.5.1.3 GTMDI development status register (GTMDI_DS)

The GTMDI_DS register, described in the following table, shows the status of various conditions that impact GTM development support. All status bits are dynamic and do not require clearing. Debug Status Event is queued due to a change in the Halt or Stop modes and a Debug Status Message is generated as soon as there is an opportunity in the NAR interface. If any status bit changes before the Status message is sent, this change is reflected in the message. The GTMDI_DS register dynamically tracks the debug condition status even when several of them are simultaneously set.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HLT	0	0	HS1												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STP	0	0	HS2												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DS register functions.

Note

The HS1 and HS2 bit fields are cleared as soon as the GTM resumes normal operation which is controlled by the CHR bit in the GTMDI_DC register. One exception is HS1[0] that indicates system debug request, which is not cleared by CHR. This bit is cleared when system debug request is negated.

Table 47-6. GTMDI_DS field descriptions

Field	Description	
31 HLT	Halt Indication. The HLT bit indicates if the GTM IP is in Halt state, ready to receive debug accesses.	
	0	GTM is running
	1	GTM is in Halt state
28–16 HS1	Halt Status1. The HS1 field shows which source generated the halt condition. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt sources that occurred immediately when the halt mode is entered.	
	0000000000000	No Trigger generated
	1xxxxxxxxxxx	TIM
	x1xxxxxxxxxx	TOM

Table continues on the next page...

Table 47-6. GTMDI_DS field descriptions (continued)

Field	Description	
	xx1xxxxxxxxx	ATOM
	xxx1xxxxxxxx	SPEA
	xxxx1xxxxxxx	SPEB
	xxxxx1xxxxxx	ARU
	xxxxxx1xxxxx	DPLL
	xxxxxxx1xxxx	MCSA
	xxxxxxxx1xxx	MCSB
	xxxxxxxxx1xx	TBU0 Watchpoint 1
	xxxxxxxxxx1x	TBU1 Watchpoint 1
	xxxxxxxxxxx1	TBU2 Watchpoint 1
	xxxxxxxxxxxx1	External through system debug enable signal
15 STP	Stop Status. Indicates if the GTM is in stop mode with no active clocks. Besides the system level stop mode mechanism, the GTM may independently be programmed for STOP mode by setting the Module Disable Control bit. Thus, the STP field indicates if the GTM is in stop mode no matter if due to the system stop mode assertion or due to the MDIS bit in the GTM module control register.	
	0	GTM not stopped
	1	GTM is stopped in low power mode
12–0 HS2	Halt Status2. Similarly to HS1 this field indicates the source for the halt condition selection 2. See Figure 47-5 for reference about source selection 1 and 2. The HS2 field shows which source generated the halt condition 2. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt condition source that occurred when the halt mode is entered.	
	0000000000000	No Trigger generated
	1xxxxxxxxxxx	TIM
	x1xxxxxxxxxx	TOM
	xx1xxxxxxxxx	ATOM
	xxx1xxxxxxxx	SPEA
	xxxx1xxxxxxx	SPEB
	xxxxx1xxxxxx	ARU
	xxxxxx1xxxxx	DPLL
	xxxxxxx1xxxx	MCSA
	xxxxxxxx1xxx	MCSB
	xxxxxxxxx1xx	TBU0 Watchpoint 2
	xxxxxxxxxx1x	TBU1 Watchpoint 2
	xxxxxxxxxxx1	TBU2 Watchpoint 2
	xxxxxxxxxxxx1	$\overline{\text{EVTI}}$

47.5.1.4 TIM watchpoint control register 1 (GTMDI_TIM_WPC1)

The GTMDI_TIM_WPC1 register, shown in the following table, controls the Watchpoint Triggers WPT issued by TIM channels. These triggers are used by the GTM internal modules in order to control GTM Halt mode entering and also at SoC level. See Figure 47-5 which describes the use of this register.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG. The SEN1 and SEN2 control bits from the GTMDI_TIM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 11

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN 1	WMC 1	0	0	0	CHSEL1		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN 2	WMC 2	0	0	0	CHSEL2		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TIM_WPC1 register functions.

Table 47-7. GTMDI_TIM_WPC1 field descriptions

Field	Description
29–28 TSS1	TIM Channel Slope Selection 1. The TSS1 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26–24 SSEL1	TIM sub-module Source Selection 1. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 TIM2 011 TIM3 100 TIM4

Table continues on the next page...

Table 47-7. GTMDI_TIM_WPC1 field descriptions (continued)

Field	Description
	101 TIM5 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
18–16 CHSEL1	TIM channel selection 1. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7
13–12 TSS2	TIM Channel Slope Selection 2. The TSS2 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10–8 SSEL2	TIM sub-module Source Selection 2. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 TIM2 011 TIM3 100 TIM4 101 TIM5 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM

Table continues on the next page...

Table 47-7. GTMDI_TIM_WPC1 field descriptions (continued)

Field	Description
	1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC1 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2–0 CHSEL2	TIM channel selection 2. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7

47.5.1.5 TIM watchpoint control register 2 (GTMDI_TIM_WPC2)

The GTMDI_TIM_WPC2 register, shown in the following table, controls the Watchpoint Triggers WPT issued by TIM channels. These watchpoint triggers are used to control GTM Halt state entering, watchpoint messages and control external logic to the GTMDI. See Figure 47-5 for details of the logic that uses the bits in this register.

Register index: 12

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TIM_WPC2 register functions.

Note

The SHARED_WPT and SHARED_WPT_ST signals mentioned in Table 47-8 and in other tables in this document, refer to the GTMDI top level connections. They correspond to Watchpoint trigger outputs of GTMDI and Stop/Start input signals respectively.

Table 47-8. GTMDI_TIM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued. This enable refers to the output selected by the WTSEL1 field. 0 Disable Watchpoint Trigger Generation 1 Enable Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TIM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same watchpoint trigger, in this case the result signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4]

Table continues on the next page...

Table 47-8. GTMDI_TIM_WPC2 field descriptions (continued)

Field	Description
	101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used to control the watchpoint trace messages for the selected TIM channel. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TIM channel. This enable refers to the output selected by the WTSEL2 field. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.6 TOM watchpoint control register 1 (GTMDI_TOM_WPC1)

The GTMDI_TOM_WPC1 register shown in the following table controls the Watchpoint Triggers issued by TOM channels. These Watchpoint triggers are used to control the GTM requests to enter Halt, issue watchpoint messages and also by logic external to the GTMDI at SoC level.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 17

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN1	WMC1	0	0	CHSEL1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN2	WMC2	0	0	CHSEL2			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TOM_WPC1 register functions.

Table 47-9. GTMDI_TOM_WPC1 field descriptions

Field	Description
29–28 TSS1	TOM Channel Slope Selection 1. The TSS1 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26–24 SSEL1	TOM sub-module Source Selection 1. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 TOM2 011 TOM3 100 TOM4 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
19–16 CHSEL1	TOM channel selection 1. Selects which channel within a selected TOM sub-module generates watchpoints. 0000 TOM CH0

Table continues on the next page...

Table 47-9. GTMDI_TOM_WPC1 field descriptions (continued)

Field	Description
	0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15
13–12 TSS2	TOM Channel Slope Selection 2. The TSS2 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10–8 SSEL2	TOM sub-module Source Selection 2. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 TOM2 011 TOM3 100 TOM4 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start and Stop signals inputs to the GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table continues on the next page...

Table 47-9. GTMDI_TOM_WPC1 field descriptions (continued)

Field	Description
3–0 CHSEL2	TOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15

47.5.1.7 TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

The GTMDI_TOM_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by TOM channels. These watchpoints are used to force GTM into Halt state, generate watchpoint messages and control logic external to the GTMDI.

Register index: 18

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TOM_WPC2 register functions.

Table 47-10. GTMDI_TOM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the TOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the TOM selected channel. Other modules can select the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1]

Table continues on the next page...

Table 47-10. GTMDI_TOM_WPC2 field descriptions (continued)

Field	Description
	10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the TOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.8 ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

The GTMDI_ATOM_WPC1 register, shown in the following table, controls the Watchpoint Triggers issued by ATOM channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ATOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 23

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		SSEL1				HEN1	WMC1	0	0	0	CHSEL1		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TSS2		SSEL2				HEN2	WMC2	0	0	0	CHSEL2		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ATOM_WPC1 register functions.

Table 47-11. GTMDI_ATOM_WPC1 field descriptions

Field	Description
29–28 TSS1	ATOM Channel Slope Selection 1. The TSS1 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
27–24 SSEL1	ATOM sub-module Source Selection 1. Selects which ATOM sub-module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 0011 ATOM3 0100 ATOM4 0101 ATOM5 0110 ATOM6 0111 ATOM7 1000 ATOM8 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected ATOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected ATOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
18–16	ATOM channel selection 1. Selects which channel in an ATOM module generates watchpoints.

Table continues on the next page...

Table 47-11. GTMDI_ATOM_WPC1 field descriptions (continued)

Field	Description
CHSEL1	000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7
13–12 TSS2	ATOM Channel Slope Selection 2. The TSS2 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
11–8 SSEL2	ATOM sub-module Source Selection 2. Selects which ATOM module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 0011 ATOM3 0100 ATOM4 0101 ATOM5 0110 ATOM6 0111 ATOM7 1000 ATOM8 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ATOM selected channel is enabled to HALT the GTM module. If this bit is set and an event occurs on the ATOM selected channel, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled

Table continues on the next page...

Table 47-11. GTMDI_ATOM_WPC1 field descriptions (continued)

Field	Description
	1 Watchpoint Messages enabled
2–0 CHSEL2	ATOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7

47.5.1.9 ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

The GTMDI_ATOM_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by ATOM channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 24

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ATOM_WPC2 register functions.

Table 47-12. GTMDI_ATOM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected ATOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the ATOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected ATOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the ATOM selected channel. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1]

Table continues on the next page...

Table 47-12. GTMDI_ATOM_WPC2 field descriptions (continued)

Field	Description
	10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected ATOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the ATOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.10 SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1)

The GTMDI_SPEA_WPC1 register, shown in the following table, controls the Watchpoint Triggers issued by SPEA channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SEP0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 29

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_SPEA_WPC1 register functions.

Table 47-13. GTMDI_SPEA_WPC1 field descriptions

Field	Description
26-24 SSEL	SPEA sub-module Source Selection. Selects which SPE sub-module is the source for watchpoint generation. 000 SPE0 001 SPE1 010 SPE2 011 SPE3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	SPEA NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEA NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
5–4 TSS2	SPEA DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope)

Table continues on the next page...

Table 47-13. GTMDI_SPEA_WPC1 field descriptions (continued)

Field	Description
	10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEA DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

47.5.1.11 SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

The GTMDI_SPEA_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by SPEA. These watchpoints are used for GTM to enter Halt state, generate watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Register index: 30

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_SPEA_WPC2 register functions.

Table 47-14. GTMDI_SPEA_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEA signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1]

Table continues on the next page...

Table 47-14. GTMDI_SPEA_WPC2 field descriptions (continued)

Field	Description
	010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEA NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEA NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEA signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEA DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation

Table continues on the next page...

Table 47-14. GTMDI_SPEA_WPC2 field descriptions (continued)

Field	Description
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEA DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.12 SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1)

The GTMDI_SPEB_WPC1 register, shown in the following table, controls the Watchpoint Triggers issued by SPEB channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SPEB_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 35

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_SPEB_WPC1 register functions.

Table 47-15. GTMDI_SPEB_WPC1 field descriptions

Field	Description
26-24 SSEL	SPEA sub-module Source Selection. Selects which SPE sub-module is the source for watchpoint generation. 000 SPE0 001 SPE1 010 SPE2 011 SPE3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	SPEB NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEB NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
5–4 TSS2	SPEB DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEB DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

47.5.1.13 SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

The GTMDI_SPEB_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by SPEB. These watchpoints are used for GTM to enter Halt state, generate watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Register index: 36

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_SPEB_WPC2 register functions.

Table 47-16. GTMDI_SPEB_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEB signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2]

Table continues on the next page...

Table 47-16. GTMDI_SPEB_WPC2 field descriptions (continued)

Field	Description
	11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEB NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEB NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEB signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEB DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEB DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.14 DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)

The GTMDI_DPLL_WPC1 register, shown in the following table, controls various trace and debug features of the DPLL modules as described below.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_DPLL_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 41

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	TSEL1		0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSEL2B		0	0	TSEL2A		0	WMC2	0	0	0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DPLL_WPC1 register functions.

Table 47-17. GTMDI_DPLL_WPC1 field descriptions

Field	Description
25–24 TSEL1	Watchpoint Selection 1. The TSEL1[1:0] field selects if a watchpoint trigger is issued based on TASI or SASI events. 00 Watchpoint on TASI event 01 Watchpoint on SASI event 10 Reserved 11 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on DPLL SASI or TASI to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table continues on the next page...

Table 47-17. GTMDI_DPLL_WPC1 field descriptions (continued)

Field	Description
21–20 TSS1	TASI/SASI Slope Selection. The TSS1 field selects the slope for TASI or SASI depending on TSEL1 selection. The selected slope is monitored by watchpoints. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	The HEN1 Halt Enable 1 bit controls if the DPLL SASI or TASI selected signal is enabled to HALT the GTM module. If this bit is set and an event occurs on the DPLL selected signal, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
13–12 TSEL2B	Watchpoint Selection 2B. The TSEL2B field selects the RAM access type to mach on 00 Watchpoint is issued on any access (read or write) 01 Watchpoint is issued based on write accesses 10 Watchpoint is issued based on read accesses 11 Reserved
9–8 TSEL2A	Watchpoint Selection 2A. The TSEL2A field selects if a watchpoint is issued based on the RAM1a, RAM1bc or RAM2 access. 00 Watchpoint is issued based on RAM1a access 01 Watchpoint is issued based on RAM1b access 10 Watchpoint is issued based on RAM2 access 11 Reserved
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables selected accesses on DPLL RAMs to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
1 HEN2	Halt Enable 2. The HEN Halt Enable bit controls if watchpoint from TSEL2 selection is enabled to HALT the GTM module. If this bit is set and an event occurs on an DPLL selected RAM, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM

47.5.1.15 DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

The GTMDI_DPLL_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by DPLL. These triggers are used for GTM to enter Halt state and by GTMDI external logic at SoC level. This register also controls the Watchpoint Trace messages WPM.

Register index: 42

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DPLL_WPC2 register functions.

Table 47-18. GTMDI_DPLL_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL SASI or TASI signal. Other modules can select this same WTSEL1 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages related to DPLL SASI or TASI signals. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the DPLL TASI or SASI selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the DPLL TASI or SASI selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table continues on the next page...

Table 47-18. GTMDI_DPLL_WPC2 field descriptions (continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL RAM access signal. Other modules can select this same WTSEL2 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the watchpoint trace messages related to DPLL RAM access events for RAM1A, RAM1B and RAM2. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the DPLL RAM selected access. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the DPLL RAM selected access event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.16 DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

The GTMDI_DPLL_WPC3 register, shown in the following table, defines the data value observed in the selected DPLL RAM access. A match to this value causes a Watchpoint.

Register index: 43

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DMA SK	0	0	0	0	RAM_DATA[23:16]							
W																

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RAM_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DPLL_WPC3 register functions.

Table 47-19. GTMDI_DPLL_WPC3 field descriptions

Field	Description
28 DMASK	Data Mask. The DMASK field is a mask for the data comparator. If set, the data read or written to the memory should not be considered for watchpoint generation. 0 RAM_DATA should be compared with data from RAM access 1 RAM_DATA is masked and should not be compared
23–0 RAM_DATA	Data value. The RAM_DATA[23:0] field defines the data value that is compared against the data accessed from the selected DPLL RAM.

47.5.1.17 DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

The GTMDI_DPLL_WPC4 register, shown in the following table, defines the RAM address mask for the DPLL selected RAM address comparisson.

Register index: 44

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR_MASK											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DPLL_WPC4 register functions.

Table 47-20. GTMDI_DPLL_WPC4 field descriptions

Field	Description
11–0 RAM_ADDR_MASK	RAM Address mask. The RAM_ADDR_MASK[11:0] field is a mask applied bit by bit to the RAM_ADDR defined in the GTMDI_DPLL_WPC5 register. If mask is 1, the corresponding RAM_ADDR bit is not valid for comparison, thus it always returns a match when compared to the selected RAM address.

47.5.1.18 DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

The GTMDI_DPLL_WPC5 register, shown in the following table, defines the address value that is monitored during DPLL selected RAM access. Once this address value matches with the DPLL RAM address a Watchpoint is issued.

Register index: 45

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_DPLL_WPC5 register functions.

Table 47-21. GTMDI_DPLL_WPC5 field descriptions

Field	Description
11–0 RAM_ADDR	RAM Address Compare. The RAM_ADDR[11:0] field defines the address value that is compared against the address for write and read DPLL RAM accesses. Before the comparison is executed the RAM_ADDR_MASK should be considered thus masking corresponding bits. A masked bit is always considered a match. **Note:** DPLL module has three RAM modules with different address widths. The address value in the RAM_ADDR field and the RAM address should be right aligned in order to be compared.

47.5.1.19 DPLL data trace control register (GTMDI_DPLL_DTC)

The GTMDI_DPLL_DTC register, shown in the following table, controls data trace message transmission.

Register index: 46

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWC		0	0	RAMSEL		0	DMC	STSEL		0	0	0	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the DMC control bit thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit from the GTMDI_DPLL_DTC register need to be set in order to allow external Start/Stop signals to control the DMC bit.

This table describes the GTMDI_DPLL_DTC register functions.

Table 47-22. GTMDI_DPLL_DTC field descriptions

Field	Description
13–12 RWC	Read/Write Control. This bit controls DPLL data trace message transmission. 00 Transmits only memory reads 01 Transmits only memory writes 10 Transmits all memory accesses 11 Reserved
9–8 RAMSEL	RAM Module Selection. This field selects a DPLL memory module for data tracing. 00 RAM1a selected for data tracing 01 RAM1b selected for data tracing 10 RAM2 selected for data tracing 11 Reserved
6 DMC	Data Trace Message Control. This bit is written by JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Data Trace Messages disabled

Table continues on the next page...

Table 47-22. GTMDI_DPLL_DTC field descriptions (continued)

Field	Description
	1 Data Trace Messages enabled
5–4 STSEL	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
0 SEN	Start/Stop Enable. The SEN bit field enables the DPLL data trace message generator to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.20 ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

The GTMDI_ARU_WPC1 register, shown in the following table, controls the Watchpoint Triggers issued by ARU Debugging Channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ARU_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 50

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	TSS1	0	0	0	0	HEN1	WMC 1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TSS2	0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_WPC1 register functions.

Table 47-23. GTMDI_ARU_WPC1 field descriptions

Field	Description
28 TSS1	ARU Debugging Channel 0 Data Activity Selection 1. The TSS1 bit selects the type of data activity to monitor on ARU Debugging Channel 0. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel 0 1 Compare data with expected value
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the ARU Debugging Channel 0 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 0, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches on the ARU Debugging Channel 0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
12 TSS2	ARU Debugging Channel 1 Data Activity Selection. The TSS2 bit selects the type of data activity to monitor on the ARU Debugging Channel 1. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel Monitor 1 1 Compare data with expected value
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ARU debugging Channel 1 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match on the ARU Debugging Channel 1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

47.5.1.21 ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

The GTMDI_ARU_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by ARU Debugging Channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 51

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_WPC2 register functions.

Table 47-24. GTMDI_ARU_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 0. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages for ARU Debugging Channel 0. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the ARU debugging Channel 0.

Table continues on the next page...

Table 47-24. GTMDI_ARU_WPC2 field descriptions (continued)

Field	Description
	0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the ARU Debugging Channel 0 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 1. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control for ARU Debugging Channel 1. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the ARU Debugging Channel 1 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the ARU Debugging Channel 1 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.22 ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

The ARU Watchpoint DATA0H high word register, shown in the following table, is used for comparison against the high word of the ARU Debugging Channel 0. The address of the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

Register index: 52

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA0H[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA0H[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_DATA0H register functions.

Table 47-25. GTMDI_ARU_DATA0H field descriptions

Field	Description
28–0 DATA0H	Watchpoint DATA0H. This field is compared against the high word of the ARU Debugging Channel 0.

47.5.1.23 ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

The ARU Watchpoint DATA0H low word register, shown in the following table, is used for comparison against the low word of the ARU Debugging Channel 0. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

Register index: 53

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA0L[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

R W	DATA0L[15:0]															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_DATA0L register functions.

Table 47-26. GTMDI_ARU_DATA0L field descriptions

Field	Description
28–0 DATA0L	Watchpoint DATA0L. This field is compared against the low word of the ARU Debugging Channel 0.

47.5.1.24 ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

The ARU Watchpoint DATA1H high word register, shown in the following table, is used for comparison against the high word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Register index: 54

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1H[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA1H[15:0]															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_DATA1H register functions.

Table 47-27. GTMDI_ARU_DATA1H field descriptions

Field	Description
28–0 DATA1H	Watchpoint DATA1H. This field is compared against the high word of the ARU Debugging Channel 1.

47.5.1.25 ARU watchpoint DATA1L register (GTMDI_ARU_DATA1L)

The ARU Watchpoint DATA1L low word register, shown in the following table, is used to be compared against the low word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Register index: 55

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1L[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1L[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_DATA1L register functions.

Table 47-28. GTMDI_ARU_DATA1L field descriptions

Field	Description
28–0 DATA1L	Watchpoint DATA1L. This field is compared against the low word of the ARU Debugging Channel 1.

47.5.1.26 ARU data trace control register (GTMDI_ARU_DTC)

The GTMDI_ARU_DTC register, shown in the following table, controls the ARU data trace message transmission. The GTM Module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1. The data trace for these two channels is controlled by DMC1/2 bits in this register. Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the DMC1 and DMC2 control bits thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC1 or DMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits need to be set in order to allow external Start/Stop signals to control the DMC1/2 bits.

Register index: 56

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	STSEL1		0	0	DMC 1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	STSEL2		0	0	DMC 2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_ARU_DTC register functions.

Table 47-29. GTMDI_ARU_DTC field descriptions

Field	Description
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the ARU Debugging Channel 0 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 DMC1	Data Trace Message Control. This bit controls ARU data trace messages transmission for ARU Debugging Channel 0. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signals, inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
16 SEN1	Start/Stop Enable 1. The SEN 1 bit field enables the ARU data trace message generator for ARU Debugging Channel 0 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC1 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table continues on the next page...

Table 47-29. GTMDI_ARU_DTC field descriptions (continued)

Field	Description
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the ARU Debugging Channel 1 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 DMC2	Data Trace Message Control. This bit controls ARU data trace message transmission for ARU Debugging Channel 1. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signals, inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
0 SEN2	The SEN2 bit field enables the ARU data trace message generator for ARU Debugging Channel 1 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to DMC2 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.27 MCSA/B development control register (GTMDI_MCSA_DC and GTMDI_MCSB_DC)

The GTMDI_MCSA_DC and GTMDI_MCSB_DC registers, shown in the following table, control various trace and debug features.

Note

Since MCSA and MCSB have a common sets of registers they are described together to avoiding duplication of similar information.

Register index: MCSA-60 / MCSB-77

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MCSA/B_SEL				0	0	0	0	0	0	0	0
W																

Table continues on the next page...

RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_DC and GTMDI_MCSB_DC register functions.

Table 47-30. GTMDI_MCSA_DC and GTMDI_MCSB_DC field descriptions

Field	Description
11–8 MCSA/B_SEL	MCSA/B Selection field. The MCSA/B_SEL[3:0] selection field selects which one of the MCS cores in the GTM module are assigned as MCSA/B. 0000 MCS0 0001 MCS1 0010 MCS2 0011 MCS3 0100 MCS4 0101 MCS5 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

47.5.1.28 MCSA/B watchpoint control register (GTMDI_MCSA_WPC and GTMDI_MCSB_WPC)

The MCSA/B Watchpoint Control Registers, shown in the following table, are used to configure MCSA/B watchpoints. These registers can be configured for halt/watchpoint based on instruction fetch, data read or data write, with address and data masking options.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC control bit thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC bit and Start/Stop signals occur at the same time also modifying the

same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit needs to be asserted in order to allow external Start/Stop signals to control the WMC bit.

The watchpoint channel selection is made through the GTMDI_MCSA_CE and GTMDI_MCSB_CE by the WPCE field.

Register index: MCSA-62 / MCSB-79

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	HACH		0	WMC	0	0	HME			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWO		0	0	0	0	0	EOC	HEN	0	STSEL		0	0	TEN	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_WPC and GTMDI_MCSB_WPC register functions.

Table 47-31. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions

Field	Description
25–24 HACH	Halt/Watchpoint on Active Channel. The HACH field selects which type of MCS RAM access generates a watchpoint or halt. Combining this field with HWO gives GTMDI the capability of monitoring channel activity as well as accesses to specific addresses or data. 00 Match on Instruction Fetch 01 Match on Data Read 10 Match on Data Write 11 Match on Data Read or Write
22 WMC	Watchpoint Message Control. The WMC Watchpoint Message Control bit controls the watchpoint message sent through the Message Data bus.. If this bit is asserted it enables MCSA/B Watchpoints to generate a watchpoint message This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
19–16 HME	Halt/Watchpoint Data Mask Enable. The HME field selects which data bytes are used for hardware halt/watchpoint generation when comparing to HWD register. This register field is only meaningful if HWO bit field is [x1], otherwise the data comparison is disabled independent of HME value. 0000 All bytes of data compared xxx1 Least significant byte of data masked out

Table continues on the next page...

Table 47-31. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions (continued)

Field	Description
	xx1x Second least significant byte of data masked out x1xx Second most significant byte of data masked out 1xxx Most significant byte of data masked out
15–14 HWO	Halt/Watchpoint Operand. The HWO field selects if address and/or data matching is done. Registers WPA and WPD are used for address and data comparison, respectively. If address is not compared WPA is not use. If data is not compared then WPD is not used. 00 Comparison Disabled for address or data x1 Compare data with HWD field 1x Compare address with HWA field
8 EOC	$\overline{\text{EVTO}}$ Control. The EOC bit controls the indication on the $\overline{\text{EVTO}}$ output. See MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1), MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2), MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1) and MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2) for definition about watchpoint 1 and 2. 0 Indicates Watchpoint 2 occurrence 1 Indicates Watchpoint 1 occurrence
7 HEN	Halt Enable. The HEN Halt Enable bit controls if the MCSA/B Watchpoint is enabled to HALT the GTM IP module. If this bit is set and watchpoint occurs on the MCSA/B a Halt signal is generated to the GTM IP which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
5–4 STSEL	Start/Stop input signal selection. Selects the Start/Stop input that is used to control the watchpoint messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN	Watchpoint Trigger Enable.The TEN field controls if a external trigger is issued by the MCSA/B when a watchpoint occurs. 0 Disables Trigger Generation 1 Enables Trigger Generation
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Watchpoint message to consider the Start/Stop inputs for watchpoint message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC Watchpoint Trace Message Control bit. 0 Disables trace control from Start/Stop inputs 1 Enables trace control from Start/Stop inputs

47.5.1.29 MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)

The GTMDI_MCSA_PTC and GTMDI_MCSB_PTC registers, shown in the following table, allow program fetch traces to be enabled and/or disabled on the occurrence of a watchpoint. If the same watchpoint is programed to enable and disable a trace, the occurrence of the watchpoint toggles the current value of the trace flags. If one watchpoint occurs to enable trace and a different watchpoint occurs to disable trace at the same time the current value of the trace flags also toggle.

Register index: MCSA-63 / MCSB-80

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PFTS		0	0	PFTE		0	FTC	0	0	0	0	0	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_PTC and GTMDI_MCSB_PTC register functions.

Table 47-32. GTMDI_MCSA_PTC and GTMDI_MCSB_PTC field descriptions

Field	Description
13–12 PFTS	Program Fetch Trace Start. This bit field allows Program fetch trace to be enabled at a watchpoint occurrence. See MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1), MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2), MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1) and MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2) for definition about watchpoint 1 and 2. 00 No watchpoint selected (do not start Program Fetch Trace) 01 Program Fetch Trace enabled on MCSA/B Program Watchpoint 1 occurrence 10 Program Fetch Trace enabled on MCSA/B Program Watchpoint 2 occurrence 11 Reserved
9–8 PFTE	Program Fetch Trace End. This bit field allows program trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Program Fetch Trace disabled on MCSA/B Program Watchpoint 1 occurrence 10 Program Fetch Trace disabled on MCSA/B Program Watchpoint 2 occurrence 11 Reserved
6 FTC	Fetch Trace Control. This bit controls the Fetch Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Fetch access to generate a trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI 0 Fetch Messages disabled 1 Fetch Messages enabled
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Fetch message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the FTC Fetch Trace Message Control bit.

Table continues on the next page...

Table 47-32. GTMDI_MCSA_PTC and GTMDI_MCSB_PTC field descriptions (continued)

Field	Description
	0 Disable fetch trace control from Start/Stop inputs 1 Enable fetch trace control from Start/Stop inputs

47.5.1.30 MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)

The GTMDI_MCSA_DTC and GTMDI_MCSB_DTC registers, shown in the following table, control which address ranges are enabled for data trace, and if reads and/or writes are traced in that range.

Register index: MCSA-64 / MCSB-81

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RC0	RC1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWT0		0	0	RWT1		0	DTC	DTS		0	DTE		SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_DTC and GTMDI_MCSB_DTC register functions.

Table 47-33. GTMDI_MCSA_DTC and GTMDI_MCSB_DTC field descriptions

Field	Description
17 RC0	Range Control 0. Controls the range on data trace window 0. The window 0 is defined by DTAR1 register. See MCSA/B data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1). 0 Trace addresses outside (exclusive) of data trace window 0 1 Trace address inside (inclusive) of data trace window 0
16 RC1	Range Control 1. Controls the range on data trace window 1. The window 1 is defined by DTAR2 register. See MCSA/B data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2). 0 Trace addresses outside (exclusive) of data trace window 1 1 Trace address inside (inclusive) of data trace window 1
13–12 RWT0	MCSA/B Read/Write Trace Control 0. RWT0 controls whether data trace messages are generated for MCSA/B RAM read or write accesses.

Table continues on the next page...

Table 47-33. GTMDI_MCSA_DTC and GTMDI_MCSB_DTC field descriptions (continued)

Field	Description	
	00	No MCSA/B data trace messages generated for data trace window 0
	01	Enable MCSA/B data read trace for MCS window 0
	10	Enable MCSA/B data write trace for MCS window 0
	11	Enable MCSA/B data read and write trace for data trace window 0
9–8 RWT1	MCSA/B Read/Write Trace Control 1. RWT1 controls whether data trace messages are generated for MCSA/B accesses inside MCS data trace window, and if so, whether reads, writes, or both generate the data trace messages.	
	00	No MCSA/B data trace messages generated for data trace window 1
	01	Enable MCSA/B data read trace for MCS window 1
	10	Enable MCSA/B data write trace for MCS window 1
	11	Enable MCSA/B data read and write trace for data trace window 1
6 DTC	Data Trace Control. This bit controls the Data Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Data access to generate a data trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI	
	0	Data Messages disabled
	1	Data Messages enabled
5–4 DTS	Data Trace Start. This bit field allows data trace to be enabled at a watchpoint occurrence.	
	00	No watchpoint selected
	01	Data Trace enabled on MCSA/B Watchpoint 1 occurrence
	10	Data Trace enabled on MCSA/B Watchpoint 2 occurrence
	11	Reserved
2–1 DTE	Data Trace End. This bit field allows data trace to be disabled at a watchpoint occurrence.	
	00	No watchpoint selected
	01	Data Trace disabled on MCSA/B Watchpoint 1 occurrence
	10	Data Trace disabled on MCSA/B Watchpoint 2 occurrence
	11	Reserved
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Data message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DTC Data Trace Message Control bit.	
	0	Disable data trace control from Start/Stop inputs
	1	Enable data trace control from Start/Stop inputs

47.5.1.31 MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1)

The MCSA/B Watchpoint Address Registers 1, shown in the following table, are used to configure the MCSA/B halt/watchpoint hardware. They are used for data and instruction fetch accesses.

Register index: MCSA-65 / MCSB-82

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE

The HWA field represents word addresses relative to the MCS address map, not the CPU address map. This applies to GTMDI_MCSA_WPA1, GTMDI_MCSB_WPA1, GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 registers.

This table describes the GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 register functions.

Table 47-34. GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 field descriptions

Field	Description
29–16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13–0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

47.5.1.32 MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2)

The MCSA/B Watchpoint Address 2 Registers, shown in the following table, are used to configure the MCSA/B halt/watchpoint hardware. They are used for data and instruction fetch accesses.

Register index: MCSA-66 / MCSB-83

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 register functions.

Table 47-35. GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 field descriptions

Field	Description
29–16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13–0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

47.5.1.33 MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1)

The MCSA/B Halt/Watchpoint Data 1 Registers, shown in the following table, are used to configure the MCSA/B halt/watchpoint hardware. They are used for data accesses selection only.

Register index: MCSA–67 / MCSB–84

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HWD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 register functions.

Table 47-36. GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 field descriptions

Field	Description
31–0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

47.5.1.34 MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2)

The MCSA/B Watchpoint Data 2 Registers, shown in the following table, are used to configure the MCSA/B halt/watchpoint hardware. They are used for data accesses selection only.

Register index: MCSA–68 / MCSB–85

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HWD															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HWD															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 register functions.

Table 47-37. GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 field descriptions

Field	Description
31–0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data.

Table 47-37. GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 field descriptions

Field	Description
	If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

47.5.1.35 MCSA/B channel enable register (GTMDI_MCSA_CE and GTMDI_MCSB_CE)

The GTMDI_MCSA_CE and GTMDI_MCSB_CE registers, shown in the following table, enable the monitoring for 8 MCSA/B channels independently.

Register index: MCSA–69 / MCSB–86

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CFTE								CDTE							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	WPCE							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_MCSA_CE and GTMDI_MCSB_CE register functions.

Table 47-38. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions

Field	Description
7–0 CFTE	MCSA/B Channel Fetch Trace Enable. The CFTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for fetch traces. If enabled, the corresponding channel Fetches are monitored by Fetch Trace messages. Start and Stop message controls are controlled by the FTC bit in the GTMDI_MCSA/B_PTC registers, MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC). 00000000 No channel selected for trace xxxxxxx1 Channel [0] selected xxxxxx1x Channel [1] selected xxxxx1xx Channel [2] selected xxxx1xxx Channel [3] selected xxx1xxxx Channel [4] selected xx1xxxxx Channel [5] selected x1xxxxxx Channel [6] selected 1xxxxxxx Channel [7] selected 11111111 All Channels selected

Table continues on the next page...

Table 47-38. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions (continued)

Field	Description
7–0 CDTE	MCSA/B Channel Data Trace Enable. The CDTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for data trace messages. If enabled, the corresponding channel Data Accesses are monitored by Data Trace messages. Start and Stop message are controlled by the DTC bit in the GTMDI_MCSA/B_DTC registers, MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC) and by the Data trace window. 00000000 No channel selected for trace xxxxxxx1 Channel [0] selected xxxxxx1x Channel [1] selected xxxxx1xx Channel [2] selected xxxx1xxx Channel [3] selected xxx1xxxx Channel [4] selected xx1xxxxx Channel [5] selected x1xxxxxx Channel [6] selected 1xxxxxxx Channel [7] selected 11111111 All Channels selected
7–0 WPCE	MCSA/B Channel Enable. The WPCE[7:0] is a one hot bit field which defines the channel or channels to be monitored for watchpoint, data and fetch traces. The messages controlled by this bit must be also enabled and disabled according to occurrence of Watchpoints and Address Ranges. 00000000 No channel selected for trace xxxxxxx1 Channel [0] selected xxxxxx1x Channel [1] selected xxxxx1xx Channel [2] selected xxxx1xxx Channel [3] selected xxx1xxxx Channel [4] selected xx1xxxxx Channel [5] selected x1xxxxxx Channel [6] selected 1xxxxxxx Channel [7] selected 11111111 All Channels selected

47.5.1.36 MCSA/B data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1)

The GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 registers, shown in the following table, define a data trace address range. The address range is used by the MCSA/B and enables the range for reads and/or writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 0.

Register index: MCSA–71 / MCSB–88

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE

The DTSA and DTEA fields represent word addresses relative to the MCS address map, not the CPU address.

This table describes the GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 register functions.

Table 47-39. GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 field descriptions

Field	Description
29–18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA/B data trace window.
13–2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA/B data trace window.

47.5.1.37 MCSA/B data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2)

The GTMDI_MCSA_DTAR2 and GTMDI_MCSA_DTAR2 registers, shown in the following table, define a data trace address range. The address range is used by the MCSA/B and enables the range for reads and/or writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 1.

Register index: MCSA–72 / MCSB–89

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note

DTSA and DTEA fields represent word addresses which are included in the address range.

This table describes the GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR12 register functions.

Table 47-40. GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2 field descriptions

Field	Description
29–18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA/B data trace window.
13–2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA/B data trace window.

47.5.1.38 TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

The GTMDI_TBU0_WPC1 register, shown in the following table, controls the Watchpoint issued by TBU0 GTM Time Base Units (TBU). These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 94

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TSS1			0	0	0	0	HEN 1	WMC 1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TSS2			0	0	0	0	HEN 2	WMC 2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU0_WPC1 register functions.

Table 47-41. GTMDI_TBU0_WPC1 field descriptions

Field	Description
30–28 TSS1	Only bits from 26 to 3 from register GTMDI_TBU0_DATA are used for comparison TBU0 Watchpoint 1 Compare Selection. The TSS1[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data[1] 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU0 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
14–12 TSS2	TBU0 Watchpoint 2 Compare Selection 2. The TSS2[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data[1] 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved

Table continues on the next page...

Table 47-41. GTMDI_TBU0_WPC1 field descriptions (continued)

Field	Description
	101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU0 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

1. Only bits from 26 to 3 from register GTMDI_TBU0_DATA are used for comparison

47.5.1.39 TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

The GTMDI_TBU0_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by TBU0. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 95

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU0_WPC2 register functions.

Table 47-42. GTMDI_TBU0_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU0 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2]

Table continues on the next page...

Table 47-42. GTMDI_TBU0_WPC2 field descriptions (continued)

Field	Description
	11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU0 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.40 TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

The TBU0 Watchpoint DATA, shown in the following table, is used for watchpoints generation for matching with the TBU0 value.

Register index: 96

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	TBU0_DATA[26:16]										
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU0_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU0_DATA register functions.

Table 47-43. GTMDI_TBU0_DATA field descriptions

Field	Description
26–0 TBU0_DATA	TBU0 Watchpoint DATA. The field TBU0_DATA is used to compare against the TBU0 data.

47.5.1.41 TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

The GTMDI_TBU1_WPC1 register, shown in the following table, controls the Watchpoint issued by TBU1 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU1_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 101

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU1_WPC1 register functions.

Table 47-44. GTMDI_TBU1_WPC1 field descriptions

Field	Description
29–28 TSS1	TBU1 Watchpoint 1 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU1 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1 a Halt signal is generated to the GTM which enters debug state.

Table continues on the next page...

Table 47-44. GTMDI_TBU1_WPC1 field descriptions (continued)

Field	Description
	0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13–12 TSS2	TBU1 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU1 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

47.5.1.42 TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

The GTMDI_TBU1_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by TBU1. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 102

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU1_WPC2 register functions.

Table 47-45. GTMDI_TBU1_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU1 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4]

Table continues on the next page...

Table 47-45. GTMDI_TBU1_WPC2 field descriptions (continued)

Field	Description
	101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU1 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.43 TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

The TBU1 Watchpoint DATA, shown in the following table, is used for watchpoints generation for matching with the TBU1 value.

Register index: 103

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU1_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU1_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU1_DATA register functions.

Table 47-46. GTMDI_TBU1_DATA field descriptions

Field	Description
23–0 TBU1_DATA	TBU1 Watchpoint DATA. The field TBU1_DATA[23:0] is used to compare against the TBU1 data.

47.5.1.44 TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

The GTMDI_TBU2_WPC1 register, shown in the following table, controls the Watchpoint issued by TBU2 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note

Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU2_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 108

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU2_WPC1 register functions.

Table 47-47. GTMDI_TBU2_WPC1 field descriptions

Field	Description
29–28 TSS1	TBU2 Watchpoint 2 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU2 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13–12 TSS2	TBU2 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU2 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

47.5.1.45 TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

The GTMDI_TBU2_WPC2 register, shown in the following table, controls the Watchpoint Triggers issued by TBU2. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 109

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU2_WPC2 register functions.

Table 47-48. GTMDI_TBU2_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU2 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources.

Table continues on the next page...

Table 47-48. GTMDI_TBU2_WPC2 field descriptions (continued)

Field	Description
	000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU2 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

47.5.1.46 TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

The TBU2 Watchpoint DATA, shown in the following table, is used for watchpoints generation for matching with the TBU2 value.

Register index: 110

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU2_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU2_DATA[15:0]															
W																

Table continues on the next page...

RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table describes the GTMDI_TBU2_DATA register functions.

Table 47-49. GTMDI_TBU2_DATA field descriptions

Field	Description
23–0 TBU2_DATA	TBU2 Watchpoint DATA. The field TBU2_DATA[23:0] is used to compare against the TBU2 data.

47.5.2 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 47-3 . Unimplemented registers should not be accessed. GTMDI treats unimplemented registers like the JTAG BYPASS instruction.

47.6 Functional description

This section describes in detail the main functionality of the GTMDI module.

47.6.1 GTM-IP Debug / Halt mode description

The GTM-IP debug / halt mode is requested via GTMDI and is used to debug the GTM-IP module. In this mode it is required that the GTM-IP is halted by stopping its clock. Therefore all time bases and counters are frozen.

[Covers: ADD21.010] GTM halt is only activated when no read- or write-access to GTM over AEI are taking place in parallel. Only in this condition the GTM is released to change to debug / halt mode read.

It is important to notice that any peripheral slave bus access is active in this mode. [Covers: ADD21.009] The debugger is able to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted. All registers in the register map can be accessed by the peripheral slave bus, even the RAMs because they are in the register map. Of course, the clocks for the RAMs are not stopped in debug mode.

Note

If GTM-IP is in Stop mode by the setting of GTMMCR[MDIS], the debug request is not considered until the MDIS bit is cleared.

47.6.2 Debug enable logic

In Debug mode, the GTM-IP main clock is requested to stop and the bus interface clocks remain running to access all registers and memories.In order to enable the GTM to enter Halt state, which is the module debug state, there are some registers that need to be set. The sources that can put GTM in halt state are events from TIM, TOM, ATOM, MCS and several other internal GTM modules that are selected and controlled by the GTMDI_DC register bits shown in GTMDI development control register (GTMDI_DC). The system debug request external signal is also a source of halts, as well as direct JTAG write to control register. The following figure describes the logic behind the Halt signal generated by GTMDI. See GTMDI development control register (GTMDI_DC) for a description of the register bits mentioned in Figure 47-4.

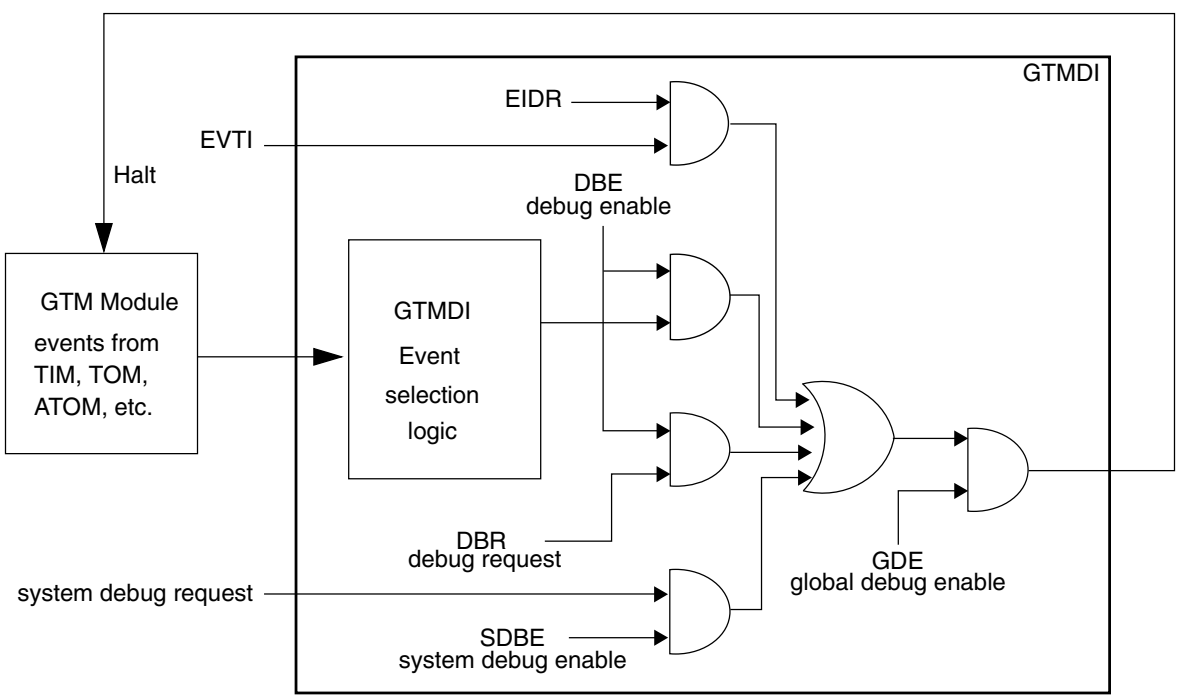


Figure 47-4. GTM Halt signal enable logic

47.6.3 TIM-TOM-ATOM selection and control logic

The TIM sub-modules can be selected to generate signals that control the watchpoint messages, Halt GTM and watchpoint outputs from the GTMDI module. The following figure describes the TIM selection logic for generation of watchpoint messages, GTM halt and external watchpoint triggers. This logic is duplicated thus allowing independent selection of two TIM filtered inputs. Figure 47-5 also shows the register bits that control the selection process. Watchpoint Logic 1 and Watchpoint Logic 2 are two implementations for the same logic.

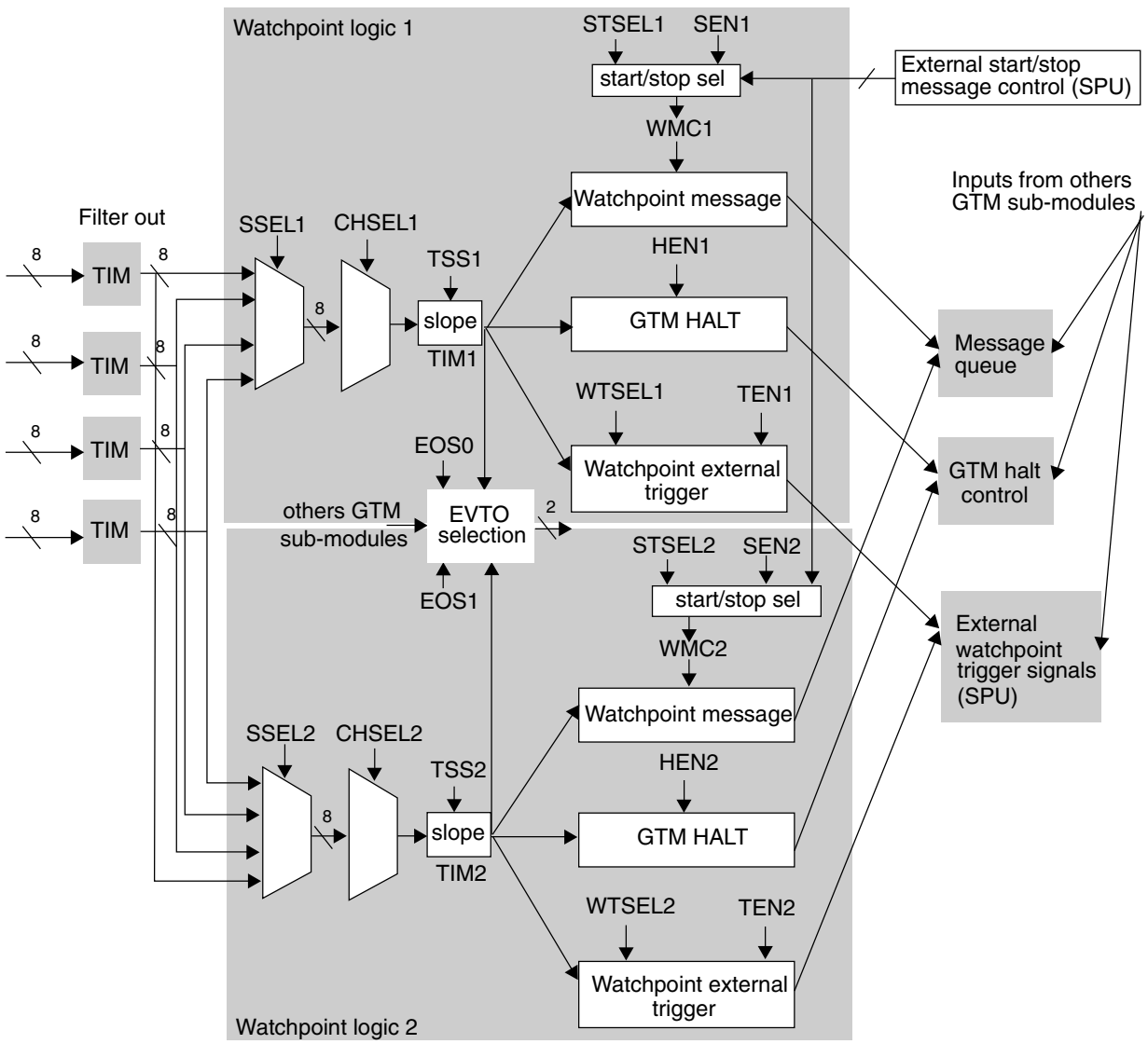


Figure 47-5. TIM selection logic

NOTE

This figure shows four instances of the TIM sub-module which corresponds to the GTM104 IP configuration. Different configurations may have different number of TIM sub-modules,

thus SSEL1/2 signals have to be able to select different number of TIM sub-modules.

The selection logic for TOM and ATOM sub-modules is similar to the TIM logic. The register names are also kept the same for easy correlation of functionality.

47.6.4 GTMDI reset configuration

This section describes the GTMDI reset configuration.

47.6.4.1 Enabling GTMDI class 1 operation

The GTMDI Class 1 features are always enabled after exiting the JTAG TEST-LOGIC-RESET state. But they are disabled while the JTAG is in this state. Thus, allowing low power mode. The registers are always reset while in the JTAG TEST-LOGIC-RESET state.

47.6.4.2 Enabling GTMDI class 3 operation

After exiting JTAG TEST-LOGIC-RESET state, the GTMDI classes 1 and 3 are enabled, otherwise the GTMDI class 3 features are disabled, entering in the Disable-Port Mode, thus no trace output is provided, and auxiliary port output pins are disabled.

47.6.5 Message Data bus—interface with NAR module

This section describes the Message Data bus.

47.6.5.1 Output messages

This table describes the messages that the GTMDI can transmit on the Message Data bus.

Table 47-50. GTMDI Nexus trace message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Debug Status Message	6	6	fixed	TCODE	Value = 0
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7

Table continues on the next page...

Table 47-50. GTMDI Nexus trace message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
	32	32	fixed	STATUS	Value of the Development Status register GTMDI_DS, see Table 47-51
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message	6	6	fixed	TCODE	Value = 8
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	4	4	fixed	ETYPE	Always 0 for GTMDI
	15	15	fixed	ECODE	Error Code, see Table 47-52
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Write with Sync Message	6	6	fixed	TCODE	Value = 56
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location written
	32	32	fixed	DATA	Data value written
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Read with Sync Message	6	6	fixed	TCODE	Value = 57
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location read
	32	32	fixed	DATA	Data value read
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message[1]	6	6	fixed	TCODE	Value = 15
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	15	15	fixed	WPHIT	Number indicating watchpoint source (binary coded), see Table 47-53, Table 47-54 and Table 47-55
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace	6	6	fixed	TCODE	Value = 58
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT[2]	Fetch Sequential Counter. Value 0 indicates there is no relationship between the last fetch trace message and the current message. The counter rolls back to one when a non sequential address is fetched. The counter increments every time a sequential address is fetched. When counter reaches 63 a message is sent and the counter rolls back to one. If an ERROR message is sent the counter rolls back to zero.

Table continues on the next page...

Table 47-50. GTMDI Nexus trace message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
	14	14	fixed	FADDR	First element address of the next fetch sequence
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace End (fetch trace was turned off, no faddr field is sent)	6	6	fixed	TCODE	Value = 59
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT	The message is sent with the current value in the FSCNT counter and the counter rolls back to zero after the message is sent.
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
ARU Data Trace	6	6	fixed	TCODE	Value = 60
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	CHN	ARU Debugging Channel number (0/1)
	58	58	fixed	DATA	Data value read {ARU_DATAH, ARU_DATAL}
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
DPLL Data Trace, Data Read/Write	6	6	fixed	TCODE	Value = 61
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	24	24	fixed	DATA	Data value read/written
	12	12	fixed	ADDR	Full address of the memory location written
	1	1	fixed	RW	Read 1, Write 0
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Watchpoint Hit Message conforms to the IEEE-ISTO 5001-2003 standard, however, other messages conform to the IEEE-ISTO 5001-2011 standard.
2. FSCNT is the sequential instruction fetch counter that indicates the number of sequential instruction fetches executed by the MCS core for a channel. See Fetch trace state machine, Fetch trace state machine for more information.

Table 47-51. Debug status field format

STATUS bit	Selected source
12:0	HS2 field of GTMDI_DS register
14:13	Reserved
15	STP bit of GTMDI_DS register
28:16	HS1 field of GTMDI_DS register
30:29	Reserved
31	HLT bit of GTMDI_DS register

Table 47-52. Error field format

ECODE bit	Selected source
2:0	TIM, TOM, ATOM

Table continues on the next page...

Table 47-52. Error field format (continued)

ECODE bit	Selected source
5:3	ARU Watchpoint, DPLL Watchpoint, SPE Watchpoint
7:6	ARU Data Trace, DPLL Data Trace
9:8	MCSA Watchpoint, MCSB Watchpoint
11:10	MCSA Data Trace, MCSB Data Trace
13:12	MCSA Fetch Trace, MCSB Fetch Trace
14	TBU Watchpoint

Table 47-53. Watchpoint field WPHIT[11:0] format for WPHIT[14:12]= 0b000

WPHIT bit	Selected source
0	ATOM 1 Negedge
1	ATOM 1 Posedge
2	ATOM 0 Negedge
3	ATOM 0 Posedge
4	TOM 1 Negedge
5	TOM 1 Posedge
6	TOM 0 Negedgee
7	TOM 0 Posedge
8	TIM 1 Negedge
9	TIM 1 Posedge
10	TIM 0 Negedge
11	TIM 0 Posedge

Table 47-54. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b001

WPHIT bit	Selected source
0	Reserved
1	Reserved
2	Reserved
3	DPLL (memory watchpoint)
4	DPLL (SASI)
5	DPLL (TASI)
6	ARU ch1
7	ARU ch0
8	MCSB1
9	MCSB0
10	MCSA1
11	MCSA0

Table 47-55. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010

WPHIT bit	Selected source
0	SPEA (DIR NEGEDGE)
1	SPEA (DIR POSEDGE)
2	SPEA (NIPD NEGEDGE)
3	SPEA (NIPD POSEDGE)
4	SPEB (DIR NEGEDGE)
5	SPEB (DIR POSEDGE)
6	SPEB (NIPD NEGEDGE)
7	SPEB (NIPD POSEDGE)
11:8	Reserved

Table 47-56. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b011

WPHIT - bit	Selected source
0	TBU0 Watchpoint 1
1	TBU0 Watchpoint 2
2	TBU1 Watchpoint 1
3	TBU1 Watchpoint 2
4	TBU2 Watchpoint 1
5	TBU2 Watchpoint 2
11:6	Reserved

47.6.5.1.1 Temporal ordering of transmitted messages

All messages are sent out in the sequence their related event actually occurred. The GTM interface with the GTMDI provides the GTM signals to be monitored and captured. At the occurrence of any event that would cause a message, a snapshot of all information needed by the message formatter to generate a message is queued. Doing so, several events from either MCS, TIM, ATOM, ARU, etc., can be queued at the same time.

47.6.5.1.2 Interleaved messages

Two MCS cores, TIM channels and other sources can be enabled for trace. The GTMDI formats and queues all messages in the order they are generated. Since there can be message inversion among sources during the send process, if enabled by the TSFT bit in GTMDI development control register (GTMDI_DC) or by the NAR, the messages are sent along with a timestamp. This timestamp is globally synchronized among all Nexus clients at SoC level.

47.6.6 IEEE 1149.1 (JTAG) input port

The GTMDI block uses the IEEE 1149.1 TAP for accessing registers. This port is shared among all TAP controllers on the SoC. This sharing is controlled by the value in the instruction register and client select control register. The GTMDI implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1 state machine shown in Figure 47-7 . The Nexus controller state machine, shown in the following table, is defined by the IEEE-ISTO 5001-2002 standard.

The instructions implemented by the GTMDI TAP controller are listed in the following table. If any other instruction is received it is treated as a BYPASS.

Table 47-57. Implemented instructions

Instruction name	Private/ public	Opcode	Description
NEXUS-ENABLE	Public	4b0000	Activate Nexus controller state machine to read and write GTMDI registers
BYPASS	Public	4b1111	Implements a single shift register stage providing a minimum serial length between TDI and TDO

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

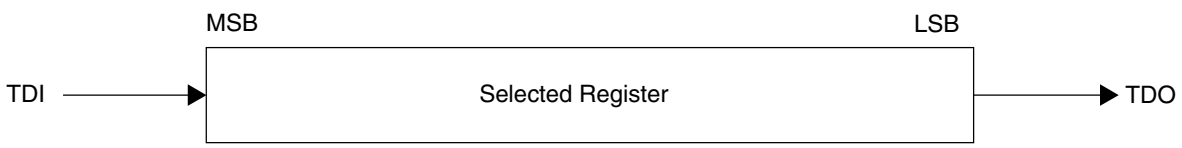


Figure 47-6. Shifting data into register

47.6.6.1 Enabling the GTMDI TAP controller

Assertion of the $\overline{\text{TRST}}$ input resets all TAP controllers on the SoC. The controllers are loaded with the BYPASS instruction upon exit of the TEST-LOGIC-RESET JTAG controller state. Loading the NEXUS-ENABLE instruction grants access to Nexus debug. GTMDI acts as BYPASS by loading any other instruction. The TDO bit is driven with the value of the instruction register while in SHIFT-IR state.

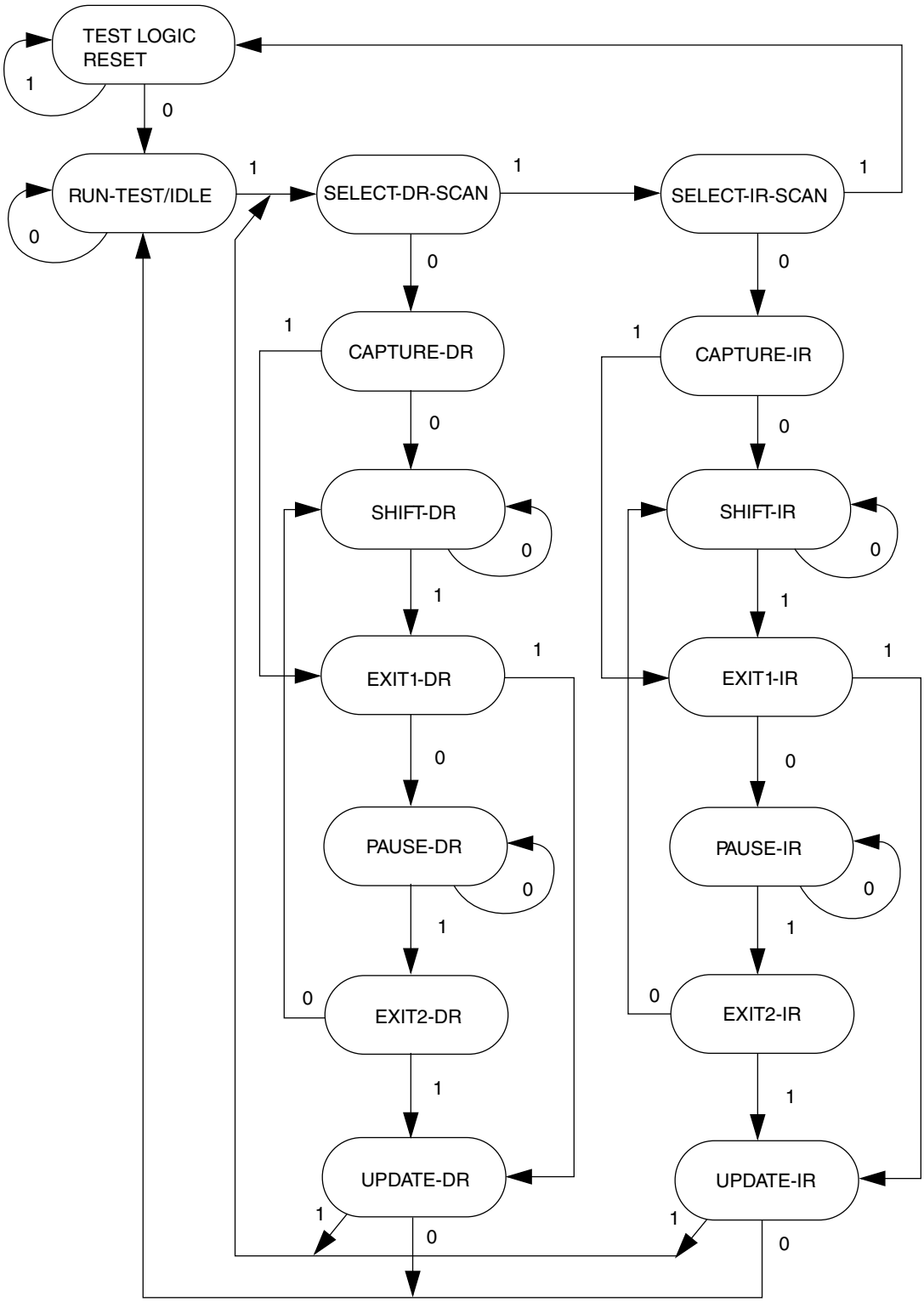


Figure 47-7. IEEE 1149.1 16-State finite state machine

47.6.6.2 Loading NEXUS-ENABLE instruction

Access to the GTMDI registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. The current instruction value is loaded into the IEEE 1149.1 shifter in the CAPTURE-IR state. The instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in the following figure, transitions to the REG_SELECT state. Table 47-58 illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

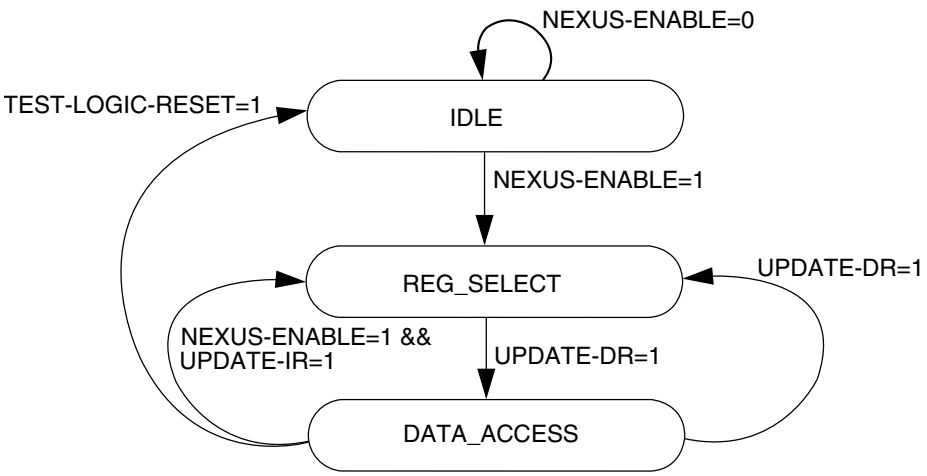


Figure 47-8. Nexus controller state machine

Table 47-58. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction
4 TCKs				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	NEXUS-ENABLE ready to be read

47.6.6.3 Selecting an IEEE 1149.1 register

When the NEXUS-ENABLE instruction is loaded in the GTMDI TAP controller instruction register, the input port allows development tool access to all GTMDI registers. Each register has a 7-bit address index. Some clients have registers with the same index as registers in other clients. Therefore, to access the registers of a particular client, the Client Select Control register must be set for that client.

All register access is performed via the SELECT-DR-SCAN path. The Nexus state machine defaults to the register select state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following figure. The read/write control bit is set to 1 for writes and 0 for reads. The current value of the Controller Command Input is captured into the IEEE 1149.1 shifter during the CAPTURE-DR state while the Nexus controller state machine is in the REG_SELECT state.

Table 47-59. IEEE 1149.1 controller command input

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the current register value is loaded into the IEEE 1149.1 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

This table illustrates a sequence which writes a 32-bit value to a register.

Table 47-60. Writing to a register

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI

Table continues on the next page...

Table 47-60. Writing to a register (continued)

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register

47.6.7 Nexus Class 1 development support

The GTMDI contains a number of hardware hooks that aid in the development of software for the GTM MCS cores. This features described in this section are compliant with the following Nexus Class 1 features:

- Enter a debug mode at reset negation
- Enter a debug mode during normal execution
- Set halts or watchpoints

Refer to Initialization/application information for how the following functions provide these features.

47.6.8 Debug status

The GTMDI module provides the Debug Status via the NAR SoC level modules.

47.6.8.1 Messaging

The GTMDI block provides debug status messaging using IEEE-ISTO 5001-2002 standard-defined public messages. If the message queue is full and a new message has to be sent, this message is lost and an ERROR message is queued instead, indicating the loss of trace messages. The debug status message format is shown in this figure.

Table 47-61. Debug status message format

6 bits	K bits	32 bits
TCODE (0)	SRC	STATUS

Length = 38 + K bits

47.6.8.2 Error messages

A debug status overrun error event is queued if a debug status event occurs while the Event Queue is full and is not able to store snapshots. The error message is stored to the event queue indicating a message was lost. The error message is sent as soon as the client is enabled for transmission. The error message format is shown in this figure.

Table 47-62. Debug status error message format

6 bits	K bits	4 bits	15 bits
TCODE (8)	SRC	ETYPE (Table 47-50)	ECODE (Table 47-52)

Length = 25 + K bits

Note

The timestamp value in an Error Message does not contain valid information.

47.6.9 Data trace

MCS cores, ARU and DPLL perform loads and stores to RAM memory modules. The GTMDI block gathers information about these accesses and generate trace messages if enabled.

47.6.9.1 MCS data write message

The MCS data write message contains the data write value and the address of the RAM target location. The data write message format is shown in the figure.

Table 47-63. Data write message format

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (56)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

47.6.9.2 MCS data read message

The MCS data read message contains the data read value and the address of the RAM target location. The data read message format is shown in this figure.

Table 47-64. Data read message format

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (57)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

47.6.9.3 MCS data trace operation

Data tracing is performed by snooping a dedicated MCS/Nexus interface for RAM read and write cycles. Data trace functions are enabled by setting the appropriate fields in the following registers:

- MCSx Development Control (GTMDI_MCSx_DC) Register
- MCSx Data Trace Control (GTMDI_MCSx_DTC) Register
- MCSx Data Trace Address Range (GTMDI_MCSx_DTAR) Register

For details on register configuration, refer to Register descriptions. The MCSx data tracing have independent Data Trace Address Range registers which allows more flexibility. An Error condition may occur in case of Queue full event. Data trace flow is depicted in this figure.

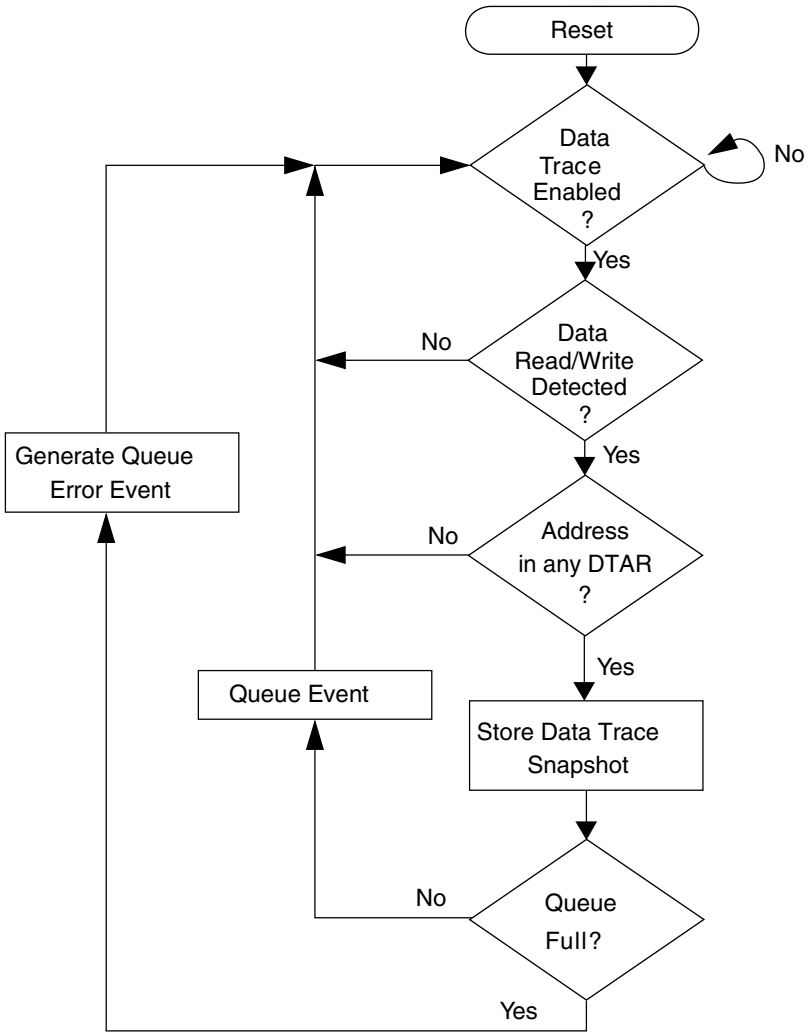


Figure 47-9. GTMDI generic data trace flow diagram

47.6.9.3.1 MCS data trace windowing

Data trace windowing is provided so that the development tool can decrease the auxiliary port usage by limiting the accesses that are traced. Data trace windowing is achieved via the address range defined by the DTSA and the DTEA fields of the DTAR register. RAM accesses are traced if their address fall in any of these ranges and the specific range is enabled in the data trace control register. Data read and/or data write trace may be enabled via the Read/Write Trace Control fields in the data trace control registers.

Data trace ranges are 32-bit aligned. This alignment is done by making the two least significant bits of the DTSA and DTEA fields read only. Since the two least significant bits of DTSA and DTEA are read only and default to different values, DTSA can never be equal to DTEA. This figure shows the relationship between the DTSA and DTEA fields.

Table 47-65. Data trace address range options

Programmed Value	Range Selected
DTSA =< DTEA	[DTSA: DTEA]
DTSA > DTEA	All addresses are out of range[1]

1. Since all addresses are considered to be out of range, a Data Trace Event may be recognized if the DTC register is programmed so that addresses out of range are queued.

Note

Before sending through the NAR interface, the MCS, DPLL or ARU Data Trace message is buffered. Due to bandwidth constraints in the NAR interface and in the data trace buffering, this data trace message can be lost when superseded by another data trace message from the same source. In this case an error message is sent. See Table 47-50 for more details about GTMDI NAR messages.

47.6.9.4 DPLL data trace

DPLL Data Trace is a straight forward implementation in which the trace is enabled using the DMC bit in the DPLL data trace control register (GTMDI_DPLL_DTC). It is possible to select individual DPLL RAM modules such as RAM1a, RAM1b or RAM2 for tracing using the RAMSEL bit in the same register. Only one memory module is selected for tracing at a given time. Start/Stop inputs from the SPU module can be used to control the DPLL Data Trace messages as well as JTAG interface. This is achieved by controlling the DMC bit which can be done from SPU or JTAG interface. See Table 47-22 for more details about the bits that should be used in this process.

47.6.9.5 ARU data trace

The GTM module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1, see the GTM documentation for details on how to program these channels. The data trace for these two channels is controlled by DMC1/2 bits in the GTMDI_ARU_DTC, see ARU data trace control register (GTMDI_ARU_DTC). Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

47.6.10 Fetch trace

Fetch trace tries to record all instruction fetch accesses done by an MCS channel. Fetch trace is handled independently for each MCS channel; therefore what happens to a certain channel does not affect other channels since all channels run independently. Even error conditions are handled independently for each channel.

Fetch trace can be enabled/disabled by two ways:

- By watchpoint hit, if the watchpoint is configured to do so
- Directly by register

When a fetch access hits a watchpoint and the last enables fetch trace (having it disabled before the watchpoint hit), that access is ignored by the fetch trace logic.

47.6.10.1 Enabling MCS program fetch trace

Program fetch trace for the MCS is enabled on a channel by channel basis. This selective enabling allows the debug tool to control the bandwidth for the transmitted messages. The program fetch trace is enabled through a JTAG register, see MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC). Since the registers may be changed in the middle of a program execution, program fetch trace may be enabled or disabled during program execution as well.

47.6.10.2 Fetch trace state machine

Fetch trace state machine is composed basically by three states: Start, Running and End. When the fetch trace is enabled, the fetch state machine goes to the Start state. This figure describes the fetch trace machine state diagram.

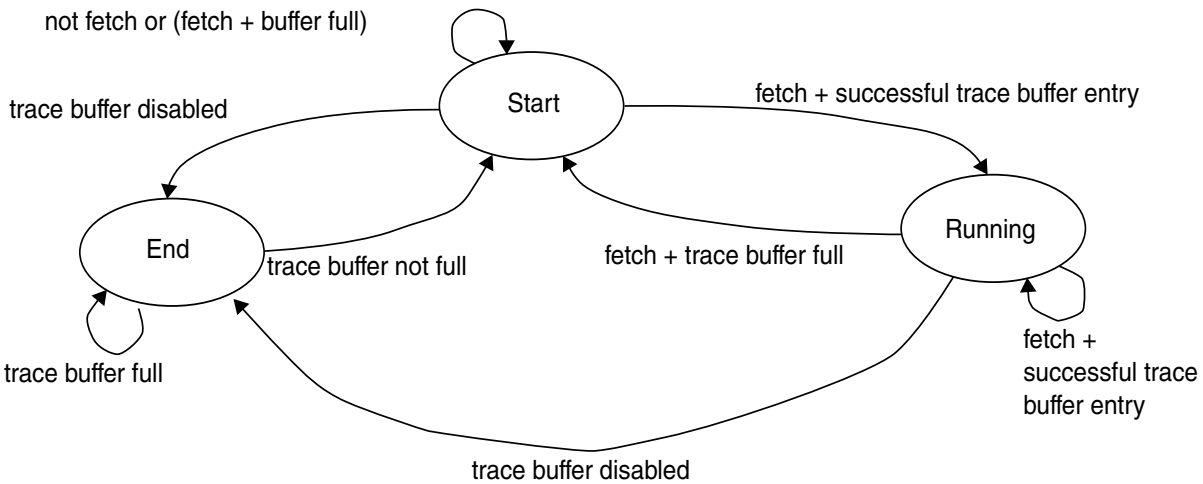


Figure 47-10. Fetch trace state machine state diagram

47.6.10.2.1 Fetch trace start state

In this state, the fetch trace logic waits for a fetch access to occur. When a fetch access occurs, the fetch trace logic tries to insert (into the corresponding MCS trace buffer) a fetch trace message with FSCNT=0 and the FADDR field equal to the fetch access address. It is possible that the corresponding MCS trace message buffer is full when fetch trace logic tries to insert the message. In this case, the fetch trace logic ignores the current fetch access, does not generate any error, and tries again when another fetch access occurs.

After inserting a message into the buffer, the fetch trace state machine goes to the Running state. It is also possible that the fetch trace is disabled before a message is inserted into the MCS trace buffer. In this case state machine goes to the End state.

47.6.10.2.2 Fetch trace running state

When the fetch trace state machine enters the Running state, a counter called CNT, is initialized to 1 and starts counting the number of sequential instruction fetch accesses.

When a fetch access with address ADDR is detected, the fetch trace logic tests if ADDR is equal to the previous fetch address, ADDR_PRV, +1 (ADDR = ADDR_PRV + 1). If so, no message is buffered for transmission and CNT increments by 1 (CNT = CNT + 1). Otherwise, if ADDR is not a sequential address, a fetch trace message is generated with the FSCNT field equals to the CNT counter value (FSCNT = CNT) and FADDR equals to the current address ADDR. The CNT counter is then initialized to 1.

At the moment the previous message is generated, the corresponding MCS fetch trace buffer may or may not be full. If the buffer is not full, the message is inserted into the buffer and the fetch trace state machine continues in the Running state, otherwise a fetch trace error message is generated for the corresponding MCS and the fetch trace state machine goes to the Start state.

NOTE

The MCS is a multi-thread core which executes eight threads, also called channels, in fixed time-slots of one system clock for each thread. Fetch trace is a per channel resource. Thus if more than one channel is selected for trace, each one generates its corresponding fetch trace message. There is one independent fetch state machine and CNT counter for each selected channel.

47.6.10.2.3 Fetch trace end state

In this state, the fetch trace logic ignores all fetch accesses and waits for the corresponding MCS trace buffer to be not full. When this happens, a fetch trace end message is inserted into the buffer with FSCNT equal to CNT (FSCNT = CNT). The CNT value is always reset to 0 when previous state was Start state.

If the buffer is always full, the fetch trace logic does not exit End state until a reset occurs.

Note

A fetch trace end message is sent when a system reset is detected in the GTM module. There are no further indications about the GTM reset condition.

47.6.11 Watchpoint trace

The GTMDI module provides watchpoint messaging via the auxiliary port, as defined by the IEEE-ISTO 5001-2002 standard.

47.6.11.1 Messaging

The GTMDI block provides watchpoint messaging using IEEE-ISTO 5001-2002 standard-defined public messages. When a watchpoint occurs, a watchpoint event is sent to the Event Queue. If the watchpoint condition occurs while the Event Queue is full, an error message is generated. The watchpoint message format is shown in this figure.

Table 47-66. Watchpoint hit message format

6 bits	K bits	15 bits
TCODE (15)	SRC	WPHIT

Length = 21+Kbits

The values for the WPHIT packet are described in Table 47-53, Table 47-54 and Table 47-55. Notice that WPHIT is never 0 because the message is only generated when a watchpoint has occurred.

47.7 Initialization/application information

This section describes the initialization and application information.

47.7.1 Accessing GTMDI tool-mapped registers

To initialize the JTAG port for register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Retrieve Device ID if needed
3. Load the Nexus TAP controller with the NEXUS-ENABLE instruction

To write control data to GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select register with a pass through the SELECT-DR-SCAN path in the JTAG state machine
2. Write the register value with a pass through the SELECT-DR-SCAN path. The prior value of this register is shifted out during the write

To read status and control data from GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear write bit to select register with a pass through SELECT-DR-SCAN path
2. Read the register value with a pass through the SELECT-DR-SCAN path. Data shifted in is ignored

See IEEE 1149.1 (JTAG) input port for more detail.

Chapter 48
CAN Subsystem

48.1 Introduction

The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules and Time triggered CAN (M_TTCAN) modules along with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection and M_TTCAN trigger memory from CPU write access. The subsystem follows the little endian format.

Note

Please refer to the Device Configuration chapter to see the number of CAN nodes used in the device.

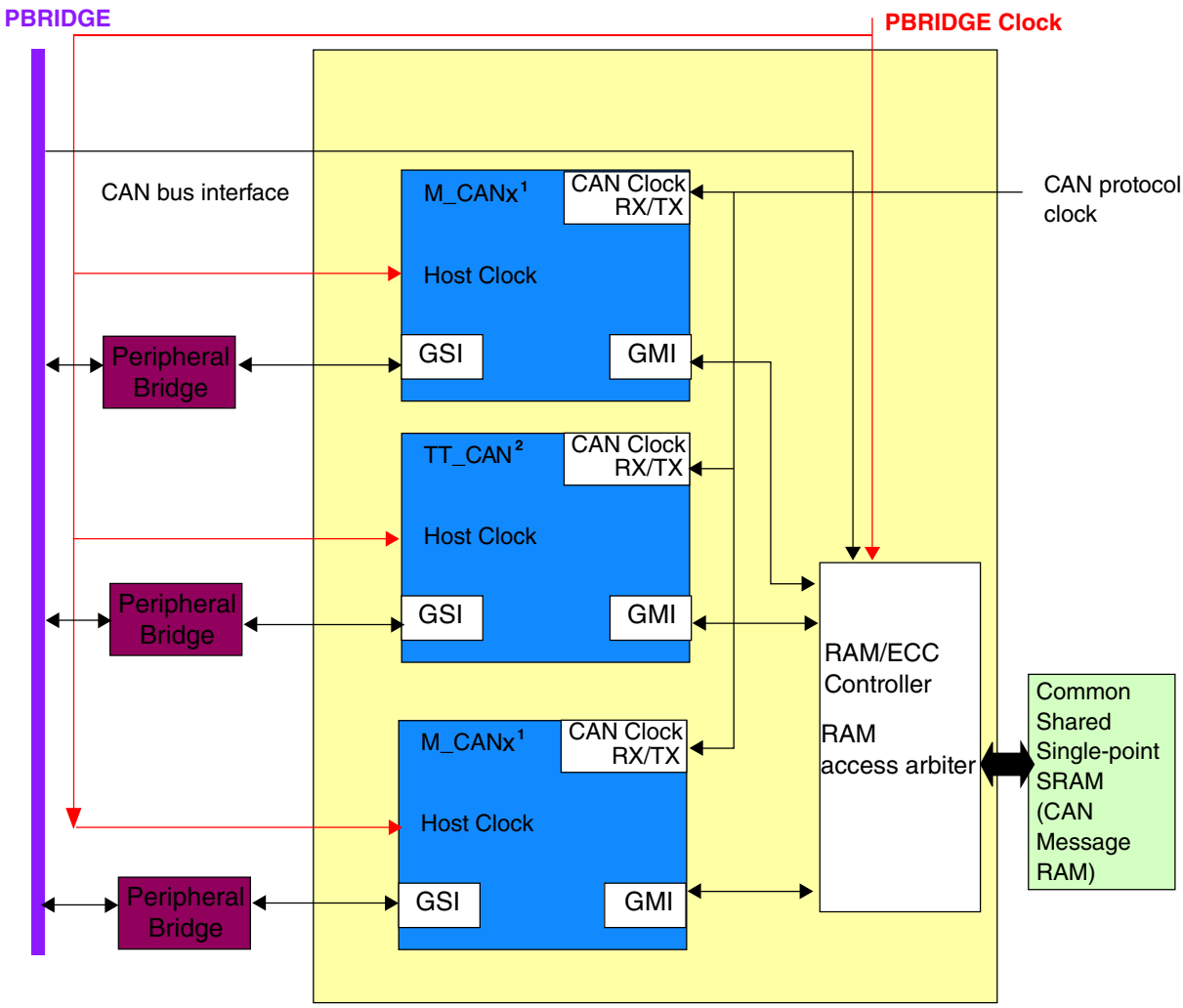
Note

M_TTCAN is implemented in a lower IP revision compared to M_CAN and therefore the supported CAN FD features are different.

Table 48-1. References

IP name	Version
M_CAN IP	Revision 3.0
M_TTCAN IP	Revision 2.0

The general CAN subsystem implementation is shown in Figure 48-1. It interfaces with the host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.



1: Number of MCAN nodes vary per device. x = 1...n. Please refer Device Configuration chapter to see the number of MCAN nodes used in the device.
2: Please refer Device Configuration chapter to see the number of M_TTCAN nodes used in the device.

Figure 48-1. CAN subsystem generic block diagram

48.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the Generic Slave Interface (GSI).
- Time triggered CAN core. The registers of the CAN module can be accessed using the Generic Slave Interface (GSI).

- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller

48.3 Modular CAN cores

CAN functionality conforms to CAN specification V2.0B active for each CAN node.The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0. Flexible assignment of Message Objects to nodes. The bit rate can be programmed to values up to 1 Mbit/s for standard CAN FD mode. High bit rates are possible in CAN FD mode. Additional transceiver hardware is required for connection to the physical layer.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen device, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

48.3.1 Features

The following are the features of Modular CAN Cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1
- CAN Flexible data-rate (CAN–FD) protocol with maximum 64 data bytes on M_CAN is supported
- Bit rates up to 1 Mbit/s in standard CAN mode

- Bit rates up to 8 Mbit/s in CAN FD mode
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug over M_CAN

48.3.2 Block diagram

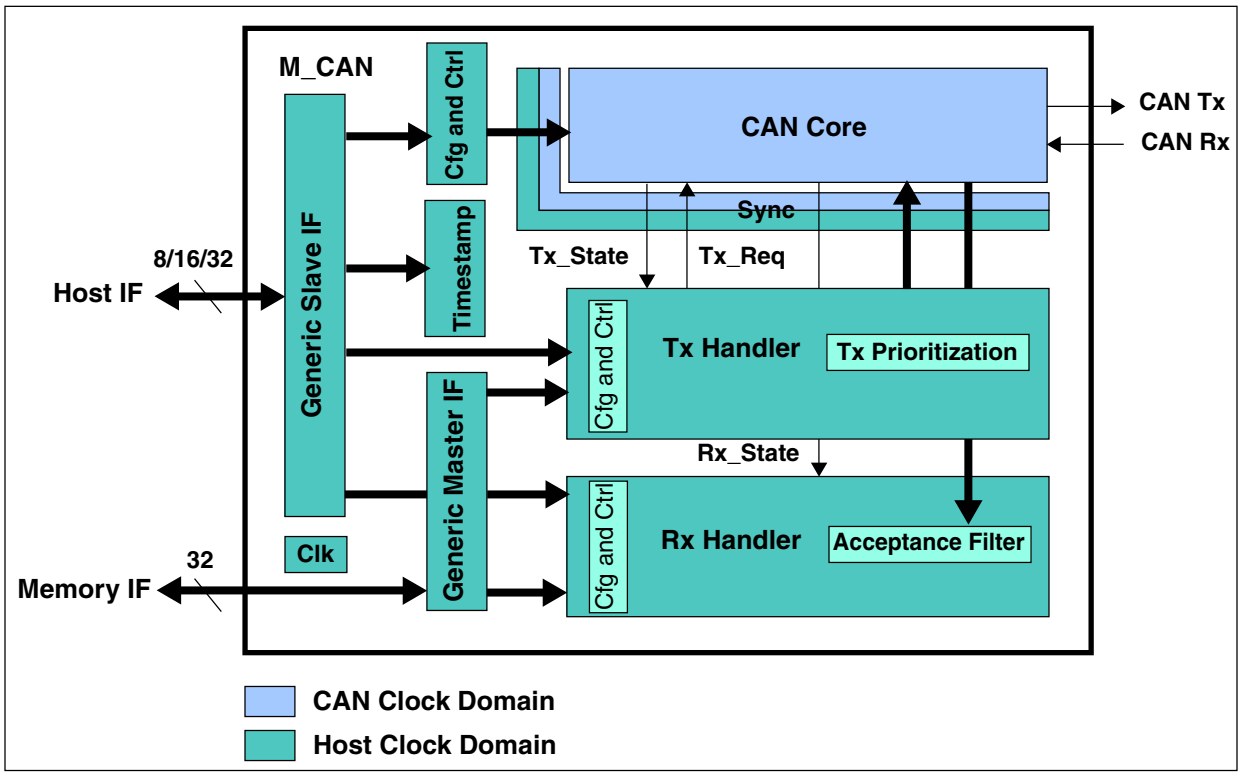


Figure 48-2. MCAN core block diagram

- CAN Core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN Core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

- Rx Handler: Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, holding all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.
- Generic Slave Interface: Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bit. Refer Table 48-37.

48.3.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note

In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

CAUTION

Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

48.3.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to EINT0 or to EINT 1. By default all interrupts are routed to interrupt line EINT 0. By programming ILE[EINT0] and ILE[EINT1], the interrupt lines can be enabled or disabled separately.

48.3.5 Memory map and register description

After hardware reset, the registers of the M_CAN hold the reset values listed in the memory map. Additionally the Bus_Off state is reset and the MCAN Tx output is set to recessive (HIGH). The value 0x0001 (bit 0 [INIT] of the CCCR = '1') in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets bit 0 [INIT] of the CCCR register = '0'.

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

NOTE

Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR is set to '1'.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

CAUTION

Any write access to reserved or not implemented registers in the 16 KB slot assigned by the peripherals bridge to the M_CAN IP will not generate any bus access error.

M_CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Core Release Register (M_CAN_CREL)	32	R	See section	48.3.5.1/ 1846
4	Endian Register (M_CAN_ENDN)	32	R	8765_4321h	48.3.5.2/ 1846
C	Fast Bit Timing and Prescaler Register (M_CAN_FBTP)	32	R/W	0000_0A33h	48.3.5.3/ 1847

Table continues on the next page...

M_CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10	Test Register (M_CAN_TEST)	32	R/W	0000_0000h	48.3.5.4/ 1849
14	RAM Watchdog Register (M_CAN_RWD)	32	R/W	0000_0000h	48.3.5.5/ 1850
18	CC Control Register (M_CAN_CCCR)	32	R/W	0000_0001h	48.3.5.6/ 1851
1C	Bit Timing and Prescaler Register (M_CAN_BTP)	32	R/W	0000_0A33h	48.3.5.7/ 1854
20	Timestamp Counter Configuration Register (M_CAN_TSCC)	32	R/W	0000_0000h	48.3.5.8/ 1855
24	Timestamp Counter Value Register (M_CAN_TSCV)	32	w1c	0000_0000h	48.3.5.9/ 1856
28	Timeout Counter Configuration Register (M_CAN_TOCC)	32	R/W	FFFF_0000h	48.3.5.10/ 1856
2C	Timeout Counter Value Register (M_CAN_TOCV)	32	w1c	0000_FFFFh	48.3.5.11/ 1857
40	Error Counter Register (M_CAN_ECR)	32	R	0000_0000h	48.3.5.12/ 1858
44	Protocol Status Register (M_CAN_PSR)	32	R	0000_0707h	48.3.5.13/ 1859
50	Interrupt Register (M_CAN_IR)	32	w1c	0000_0000h	48.3.5.14/ 1862
54	Interrupt Enable Register (M_CAN_IE)	32	R/W	0000_0000h	48.3.5.15/ 1866
58	Interrupt Line Select Register (M_CAN_ILS)	32	R/W	0000_0000h	48.3.5.16/ 1869
5C	Interrupt Line Enable Register (M_CAN_ILE)	32	R/W	0000_0000h	48.3.5.17/ 1872
80	Global Filter Configuration Register (M_CAN_GFC)	32	R/W	0000_0000h	48.3.5.18/ 1873
84	Standard ID Filter Configuration Register (M_CAN_SIDFC)	32	R/W	0000_0000h	48.3.5.19/ 1874
88	Extended ID Filter Configuration Register (M_CAN_XIDFC)	32	R/W	0000_0000h	48.3.5.20/ 1875
90	Extended ID and Mask Register (M_CAN_XIDAM)	32	R/W	1FFF_FFFFh	48.3.5.21/ 1876
94	High Priority Message Status Register (M_CAN_HPMS)	32	R	0000_0000h	48.3.5.22/ 1877
98	New Data 1 Register (M_CAN_NDAT1)	32	R/W	0000_0000h	48.3.5.23/ 1878
9C	New Data 2 Register (M_CAN_NDAT2)	32	R/W	0000_0000h	48.3.5.24/ 1878
A0	Rx FIFO 0 Configuration (M_CAN_RXF0C)	32	R/W	0000_0000h	48.3.5.25/ 1879

Table continues on the next page...

M_CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Rx FIFO 0 Status Register (M_CAN_RXF0S)	32	R	0000_0000h	48.3.5.26/ 1880
A8	Rx FIFO 0 Acknowledge Register (M_CAN_RXF0A)	32	R/W	0000_0000h	48.3.5.27/ 1881
AC	Rx Buffer Configuration Register (M_CAN_RXBC)	32	R/W	0000_0000h	48.3.5.28/ 1882
B0	Rx FIFO 1 Configuration Register (M_CAN_RXF1C)	32	R/W	0000_0000h	48.3.5.29/ 1883
B4	Rx FIFO 1 Status Register (M_CAN_RXF1S)	32	R	0000_0000h	48.3.5.30/ 1884
B8	Rx FIFO 1 Acknowledge Register (M_CAN_RXF1A)	32	R/W	0000_0000h	48.3.5.31/ 1885
BC	Rx Buffer / FIFO Element Size Configuration Register (M_CAN_RXESC)	32	R/W	0000_0000h	48.3.5.32/ 1885
C0	Tx Buffer Configuration Register (M_CAN_TXBC)	32	R/W	0000_0000h	48.3.5.33/ 1887
C4	Tx FIFO/Queue Status Register (M_CAN_TXFQS)	32	R	0000_0000h	48.3.5.34/ 1889
C8	Tx Buffer Element Size Configuration (M_CAN_TXESC)	32	R/W	0000_0000h	48.3.5.35/ 1890
CC	Tx Buffer Request Pending Register (M_CAN_TXBRP)	32	R	0000_0000h	48.3.5.36/ 1891
D0	Tx Buffer Add Request register (M_CAN_TXBAR)	32	R/W	0000_0000h	48.3.5.37/ 1892
D4	Tx Buffer Cancellation Request register (M_CAN_TXBCR)	32	R/W	0000_0000h	48.3.5.38/ 1892
D8	Tx Buffer Transmission Occurred register (M_CAN_TXBTO)	32	R	0000_0000h	48.3.5.39/ 1893
DC	Tx Buffer Cancellation Finished register (M_CAN_TXBCF)	32	R	0000_0000h	48.3.5.40/ 1893
E0	Tx Buffer Transmission Interrupt Enable register (M_CAN_TXBTIE)	32	R/W	0000_0000h	48.3.5.41/ 1894
E4	Tx Buffer Cancellation Finished Interrupt Enable register (M_CAN_TXBCIE)	32	R/W	0000_0000h	48.3.5.42/ 1894
F0	Tx Event FIFO Configuration Register (M_CAN_TXEFC)	32	R/W	0000_0000h	48.3.5.43/ 1895
F4	Tx Event FIFO Status register (M_CAN_TXEFS)	32	R	0000_0000h	48.3.5.44/ 1896
F8	Tx Event FIFO Acknowledge register (M_CAN_TXEFA)	32	R/W	0000_0000h	48.3.5.45/ 1897

48.3.5.1 Core Release Register (M_CAN_CREL)

NOTE

The coding of revisions depends on the module version used in the device.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REL				STEP				SUBSTEP				YEAR				MON								DAY							
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

M_CAN_CREL field descriptions

Field	Description
0–3 REL	Core Release. One digit, BCD.
4–7 STEP	Step of Core Release. One digit, BCD.
8–11 SUBSTEP	Sub-step of Core Release. One digit, BCD.
12–15 YEAR	Time Stamp Year. One digit, BCD.
16–23 MON	Time Stamp Month. Two digits, BCD.
24–31 DAY	Time Stamp Day. Two digits, BCD.

48.3.5.2 Endian Register (M_CAN_ENDN)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETV																															
W																																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

M_CAN_ENDN field descriptions

Field	Description
0–31 ETV	Endianness Test Value. The endianness test value is 0x87654321.

48.3.5.3 Fast Bit Timing and Prescaler Register (M_CAN_FBTP)

The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 M_CAN clock periods. tq = (FBRP + 1) MCAN clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. FTSEG2 is Phase_Seg2. Therefore the length of the bit time is (prog rammed values) [FTSEG1 + FTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

NOTE

With a M_CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.

The bit rate configured for the CAN FD data phase via FBTP must be higher or equal to the bit rate configured for the arbitration phase via BTP.

To archive high bit rates in CAN FD mode for the data phase up to 8 Mbit/s the number of time quanta bits per CAN bit can be chosen down to 4 time quanta per bit time.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			TDCO					TDC	0		FBRP				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				FTSEG1				0	FTSEG2			0		FSJW	
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

M_CAN_FBTP field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 TDCO	Transceiver Delay Compensation Offset (0x0–0x1F)— Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 31 M_CAN clock periods. **NOTE:** These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to ‘1’.
8 TDC	Transceiver Delay Compensation **NOTE:** These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to ‘1’. 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 FBRP	Fast Baud Rate Prescaler (0x000–0x1F)— The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value prog rammed here is used. **NOTE:** These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to ‘1’.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 FTSEG1	Fast time segment before sample point (0x1–0xF)— Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. **NOTE:** These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to ‘1’.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 FTSEG2	Fast time segment before sample point (0x0–0x7)— Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. **NOTE:** These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to ‘1’.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FSJW	Fast (Re) Synchronization Jump Width 0x0-0x3 Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Table continues on the next page...

M_CAN_FBTP field descriptions (continued)

Field	Description
	NOTE: These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to ‘1’.

48.3.5.4 Test Register (M_CAN_TEST)

Write access to the Test Register has to be enabled by setting CCCR[TEST] to '1'. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back mode and software control of MCAN Tx pin are hardware test modes. Programming of TX other than"00" may disturb the message transfer on the CAN bus.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		TDCV						RX	TX		LBCK	0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TEST field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_CAN_TEST field descriptions (continued)

Field	Description
18–23 TDCV	Transceiver Delay Compensation Value (0x00–0x3F)— Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx to m_can_rx and FBTP.TDCO. Valid values are 0 to 63 M_CAN core clock periods.
24 RX	Receive Pin. Monitors the actual value of pin m_ can_rx **NOTE:** The reset value of this bit is undefined 0 The CAN bus is dominant (m_can_rx = '0') 1 The CAN bus is recessive (m_can_rx = '1')
25–26 TX	Control of Transmit Pin **NOTE:** Protected write (P) bit(s) which means that the write access to this field is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'. 00 Reset value, M_CAN transmit is controlled by the CAN Core, updated at the end of the CAN bit time 01 Sample Point can be monitored at M_CAN transmit pin 10 Dominant ('0') level at M_CAN transmit pin 11 Recessive ('1') at M_CAN transmit pin
27 LBCK	Loop Back mode **NOTE:** Protected write (P) bit(s) which means that the write access to this field is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'. 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (see Test modes)
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

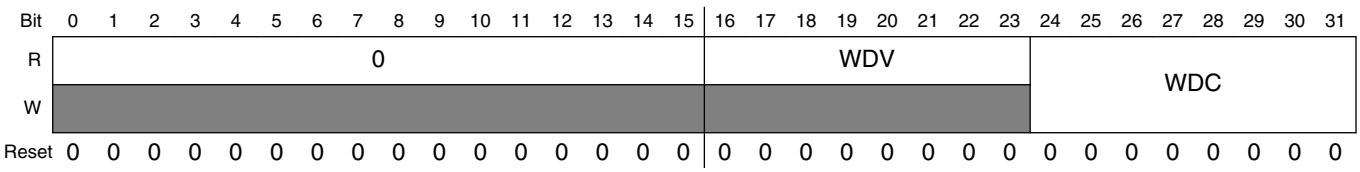
48.3.5.5 RAM Watchdog Register (M_CAN_RWD)

The RAM Watchdog monitors the READY output of the Message RAM . A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR is set. The RAM Watchdog Counter is clocked by the Host clock .

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 14h offset = 14h



M_CAN_RWD field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 WDV	Watchdog Value. Actual Message RAM Watchdog Counter Value
24–31 WDC	Watchdog Configuration. Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.

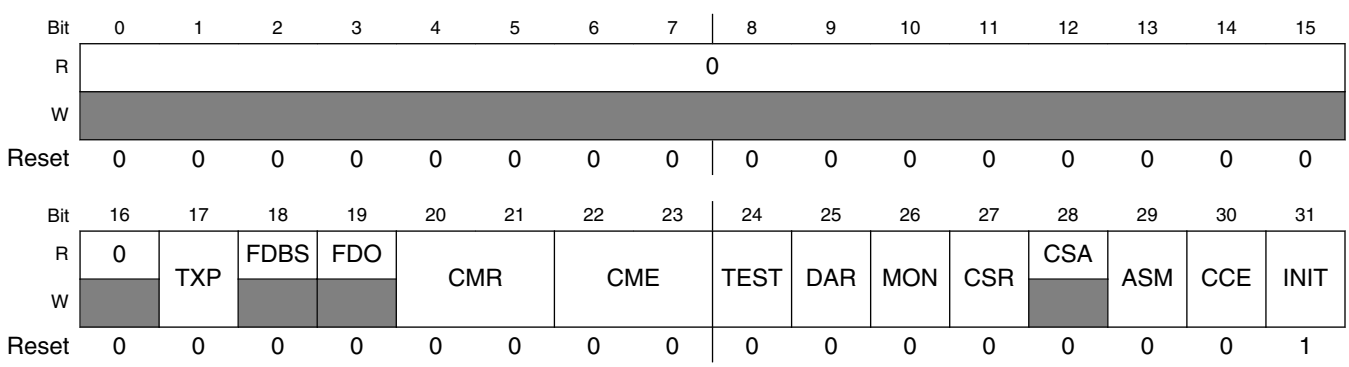
48.3.5.6 CC Control Register (M_CAN_CCCR)

This section explains the CAN Control register. For details about setting and resetting of single bits see Software initialization section.

NOTE

RWPp - Protected write and protected set register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 18h offset = 18h



M_CAN_CCCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_CAN_CCCR field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Long Frame Mode disabled, LACT may not be set 1 Long Frame Mode enabled, LACT may be set
17 TXP	Transmit Pause If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame 0 Transmit pause disabled 1 Transmit pause enabled
18 FDBS	CAN FD Bit Rate Switching 0 This node transmits no frames with bit rate switching 1 This node transmits all frames (excluding remote frames) with bit rate switching
19 FDO	Fast Frame Mode Active 0 This node transmits all frames in CAN format according to ISO11898-1 1 This node transmits all frames (excluding remote frames) in CAN FD format
20–21 CMR	CAN Mode Request A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to '00' and the status flags FDBS and FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. In case CME = '01'/'10'/'11' a change to CAN operation according to ISO 11898-1 is always possible. Default is CAN operation according to ISO11898-1. 00 unchanged 01 Request CAN FD operation 10 Request CAN FD operation with bit rate switching 11 Request CAN operation according ISO11898-1
22–23 CME	CAN Mode Enable **NOTE:** These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR are set to '1'. **NOTE:** When CME = '00', received frames are strictly interpreted according to ISO11898-1, which leads to the transmission of an error frame when receiving a CAN FD frame. In case CME = '01', transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = '10'/'11', transmission and reception of long and fast CAN FD frames is enabled 00 CAN operation according to ISO11898-1 enabled 01 CAN FD operation enabled 10 CAN FD operation with bit rate switching enabled 11 CAN FD operation with bit rate switching enabled
24 TEST	Test Enable Mode **NOTE:** This is protected set (p) bit. 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled

Table continues on the next page...

M_CAN_CCCR field descriptions (continued)

Field	Description
25 DAR	DAR: Disable Automatic Retransmission **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode. Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. **NOTE:** This is protected set (p) bit. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping host clock and core clock
29 ASM	ASM Restricted Operation Mode. Bit ASM is only set by the Host when both CCE and INIT are set to ‘1’. The bit is reset by the Host at any time. **NOTE:** This is protected set (p) bit. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization **NOTE:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value. 0 Normal Operation 1 Initialization is started

48.3.5.7 Bit Timing and Prescaler Register (M_CAN_BTP)

The CAN bit time may be programed in the range of [4....81] time quanta. The CAN time quantum may be programmed in the range of [1....1024] MCAN clock periods.

tq = (BRP + 1) MCAN clock period.

TSEG1 is the sum of Prop_Seg and Phase_Seg1. TSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

NOTE

With a CAN clock of 8 MHz, the reset value of 0x0000_0A33 configures the M_CAN for a bit rate of 500 kBit/s.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						BRP										0		TSEG1						TSEG2				SJW			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

M_CAN_BTP field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 BRP	Baud Rate Prescaler (0x000-0x3FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value prog rammed here is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TSEG1	(0x01-0x3F) Valid values are 1 to 63. The actual interpretation by the hardware of this value is such that one more than the prog rammed value is used.

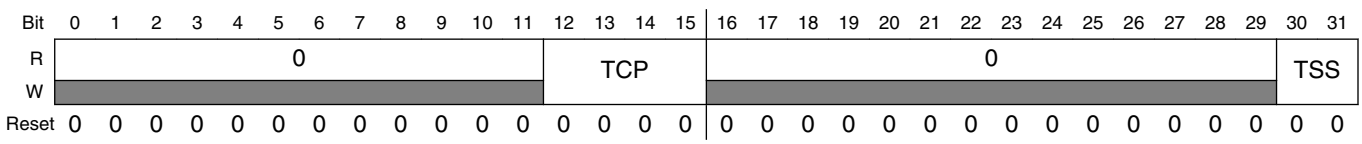
Table continues on the next page...

M_CAN_BTP field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
24–27 TSEG2	(0x0-0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the prog rammed value is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
28–31 SJW	(0x0-0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value prog rammed here is used **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.

48.3.5.8 Timestamp Counter Configuration Register (M_CAN_TSCC)

Address: 0h base + 20h offset = 20h

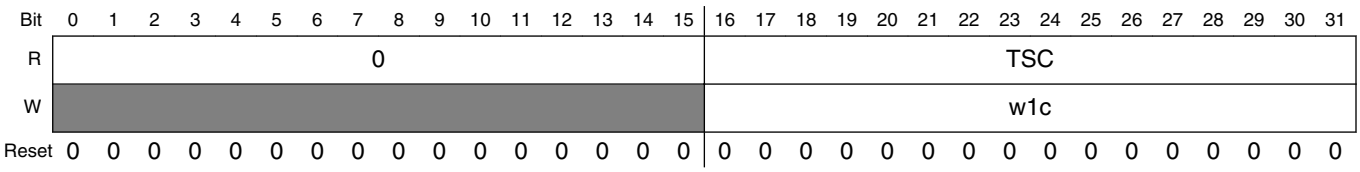


M_CAN_TSCC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 TCP	Timestamp Counter Prescaler 0x0-0xF Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase.
16–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 TSS	Timestamp Select **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to **TCP** 10 Reserved 11 Same as "00"

48.3.5.9 Timestamp Counter Value Register (M_CAN_TSCV)

Address: 0h base + 24h offset = 24h



M_CAN_TSCV field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TSC	Timestamp Counter The internal Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = '01', the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. Write access resets the counter to zero. A write access has no impact. **NOTE:** A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

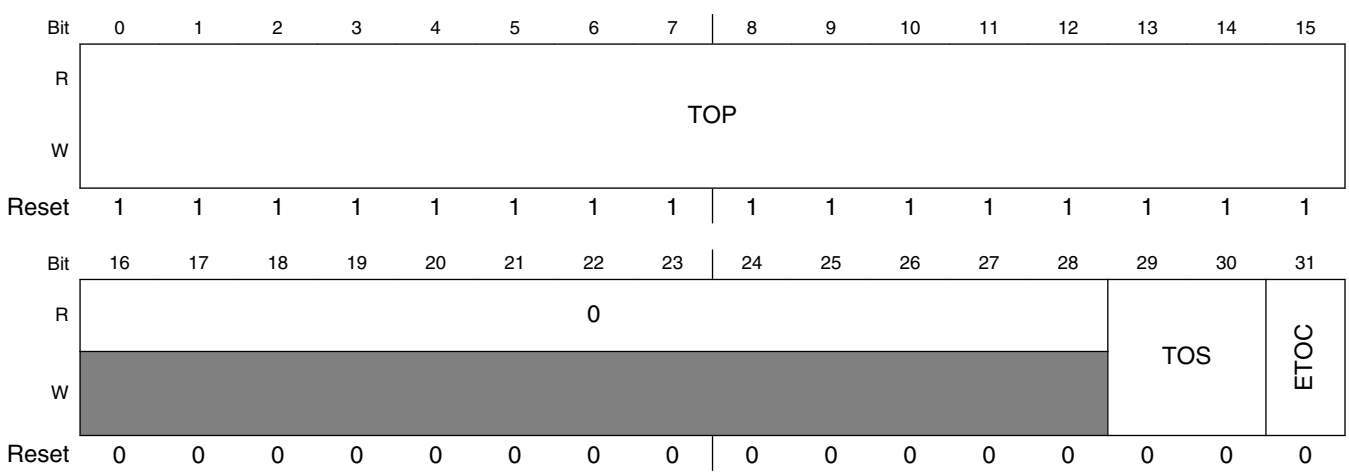
48.3.5.10 Timeout Counter Configuration Register (M_CAN_TOCC)

For a description of the Timeout Counter see Timeout counter .

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 28h offset = 28h



M_CAN_TOCC field descriptions

Field	Description
0–15 TOP	Timeout Period. Start value of the Timeout Counter (down-counter). Configures the Timeout Period. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 TOS	Timeout Select When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1 00 00 Continuous operation
31 ETOC	Enable Timeout Counter **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. **NOTE:** For use of timeout function with CAN FD see section, "Timeout Counter". 0 Timeout Counter disabled 1 Timeout Counter enabled

48.3.5.11 Timeout Counter Value Register (M_CAN_TOCV)

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TOC															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

M_CAN_TOCV field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TOC	The Timeout Counter is decremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

48.3.5.12 Error Counter Register (M_CAN_ECR)

NOTE

When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								CEL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_ECR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 CEL	CAN Error Logging. The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO bit of the IR.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17–23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127
24–31 TEC	Transmit Error Counter. Actual state of the Transmit Error Counter, values between 0 and 255.

48.3.5.13 Protocol Status Register (M_CAN_PSR)

NOTE

When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error .

NOTE

The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT.] If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 x 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		REDL	RBRS	RESI	FLEC			BO	EW	EP	ACT		LEC		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

M_CAN_PSR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_CAN_PSR field descriptions (continued)

Field	Description
18 REDL	Received CAN FD Message with EDL flag **NOTE:** This field is reset by a read operation. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with EDL flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. **NOTE:** This field is reset by a read operation. 0 Last received CAN FD message did not ha ve its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with REDL, independent of acceptance filtering. **NOTE:** This field is reset by a read operation. 0 Last received CAN FD message did not ha ve its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21–23 FLEC	Fast Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. **NOTE:** This field is set by a read operation.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state
27–28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing - node is synchronizing on CAN communication 01 Idle - node is neither receiver nor transmitter 10 Receiver - node is operating as receiver 11 Transmitter - node is operating as transmitter
29–31 LEC	Last Error Code

Table continues on the next page...

M_CAN_PSR field descriptions (continued)

Field	Description
	The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010: Form Error: A fixed format part of a received frame has the wrong format. 011: AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111: NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. **NOTE:** This field is set by a read operation.

48.3.5.14 Interrupt Register (M_CAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_IR field descriptions

Field	Description
0 STE	Stuff Error 0 No Stuff Error detected 1 More than 5 equal bits in a sequence occurred
1 FOE	Format Error 0 No Format Error detected 1 A fixed format part of a received frame has the wrong format
2 ACKE	Acknowledge Error 0 No Acknowledge Error detected 1 A transmitted message was not acknowledged by another node
3 BE	Bit Error

Table continues on the next page...

M_CAN_IR field descriptions (continued)

Field	Description
	0 No Bit Error detected 1 Device wanted to send a rec / dom level, but monitored bus level was dom / rec
4 CRCE	CRC Error 0 No CRC Error detected 1 Received CRC did not match the calculated CRC
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected. Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (e.g. parity logic)
11 BEC	Bit Error Corrected. Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (e.g. ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 MRAF	Message RAM Access Failure

Table continues on the next page...

M_CAN_IR field descriptions (continued)

Field	Description
	The flag is set, when the Rx Handler • has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. • was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler is not able to read a message from the Message RAM in time. In this case message transmission is aborted. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost

Table continues on the next page...

M_CAN_IR field descriptions (continued)

Field	Description
	0 Rx FIFO 1 not full 1 Rx FIFO 1 full
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

48.3.5.15 Interrupt Enable Register (M_CAN_IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signaled on an interrupt line

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_IE field descriptions

Field	Description
0 STEE	Stuff Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 FOEE	Format Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 ACKEE	Acknowledge Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
3 BEE	Bit Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 CRCEE	CRC Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table continues on the next page...

M_CAN_IE field descriptions (continued)

Field	Description
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable

Table continues on the next page...

M_CAN_IE field descriptions (continued)

Field	Description
	0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable

Table continues on the next page...

M_CAN_IE field descriptions (continued)

Field	Description
	0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

48.3.5.16 Interrupt Line Select Register (M_CAN_ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_ILS field descriptions

Field	Description
0 STEL	Stuff Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
1 FOEL	Format Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
2 ACKEL	Acknowledge Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

Table continues on the next page...

M_CAN_ILS field descriptions (continued)

Field	Description
3 BEL	Bit Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
4 CRCEL	CRC Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
14 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

Table continues on the next page...

M_CAN_ILS field descriptions (continued)

Field	Description
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0

Table continues on the next page...

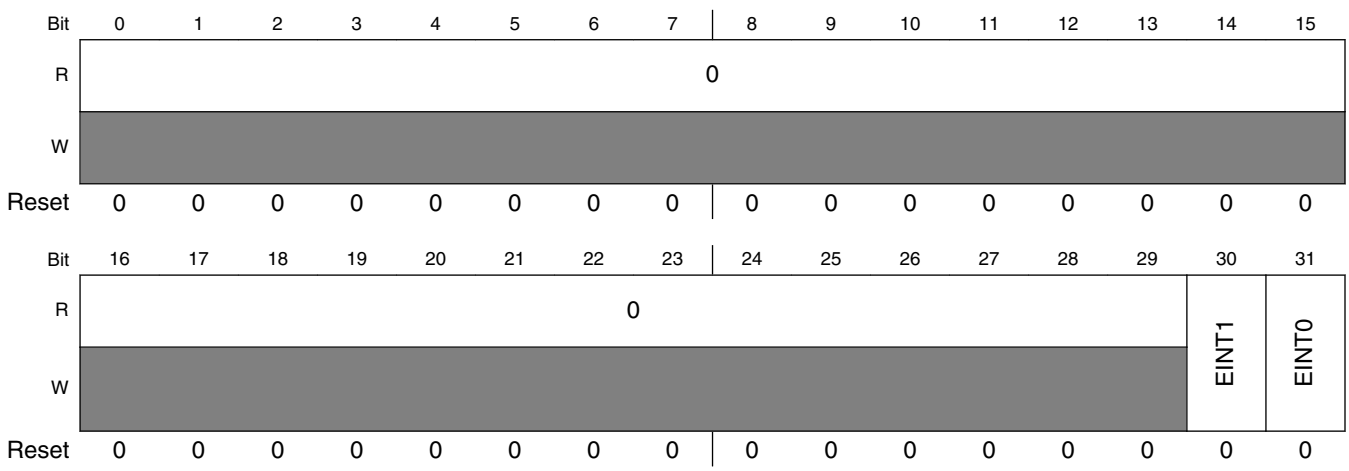
M_CAN_ILS field descriptions (continued)

Field	Description
	1 Interrupt assigned to interrupt line MCAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

48.3.5.17 Interrupt Line Enable Register (M_CAN_ILE)

Each of the two interrupt lines to the CPU can be enabled /disabled separately by programming bits EINT0 and EINT1.

Address: 0h base + 5Ch offset = 5Ch



M_CAN_ILE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled

Table continues on the next page...

M_CAN_ILE field descriptions (continued)

Field	Description
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

48.3.5.18 Global Filter Configuration Register (M_CAN_GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in the Message RAM and Extended message ID filtering .

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										ANFS		ANFE		RRFS	RRFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_GFC field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 ANFS	Accept Non-matching Frames Standard. Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28–29 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are

Table continues on the next page...

M_CAN_GFC field descriptions (continued)

Field	Description
	treated. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

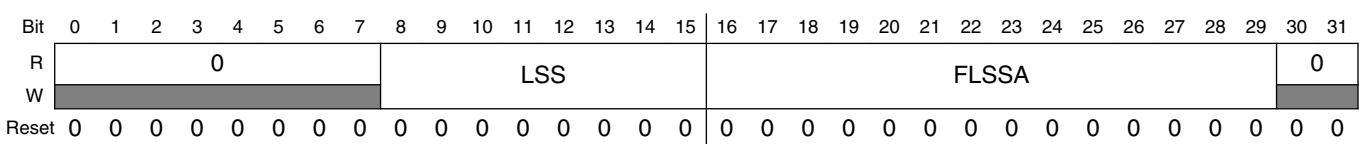
48.3.5.19 Standard ID Filter Configuration Register (M_CAN_SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for the standard messages.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 84h offset = 84h



M_CAN_SIDFC field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 LSS	List Size Standard

Table continues on the next page...

M_CAN_SIDFC field descriptions (continued)

Field	Description
	0 No standard Message ID filter 1-128 Number of standard Message ID filter elements >128 Values greater than 128 are interpreted as 128 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.20 Extended ID Filter Configuration Register (M_CAN_XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for the standard messages.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									LSE							FLESA														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_XIDFC field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LSE	List Size Extended 0 No extended Message ID filter 1-64 Number of extended Message ID filter elements >64 Values greater than 64 are interpreted as 64 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–29 FLESA	Filter List Extended Start Address

Table continues on the next page...

M_CAN_XIDFC field descriptions (continued)

Field	Description
	Start address of extended Message ID filter list (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.21 Extended ID and Mask Register (M_CAN_XIDAM)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			EIDM																												
W																																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

M_CAN_XIDAM field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.

48.3.5.22 High Priority Message Status Register (M_CAN_HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_HPMS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17–23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24–25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26–31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.

48.3.5.23 New Data 1 Register (M_CAN_NDAT1)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ND1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_NDAT1 field descriptions

Field	Description
0–31 ND1	New Data[0:31] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register.

48.3.5.24 New Data 2 Register (M_CAN_NDAT2)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ND2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_NDAT2 field descriptions

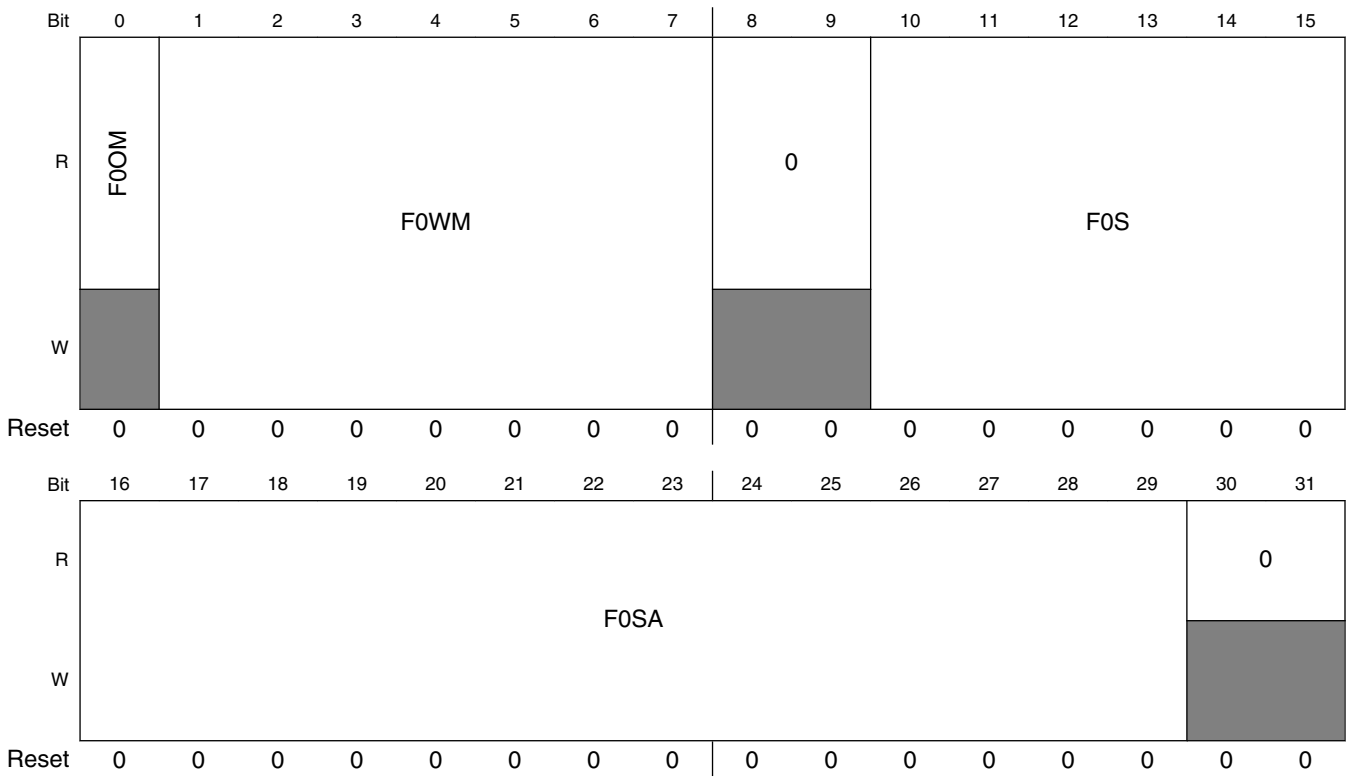
Field	Description
0–31 ND2	New Data[32:63] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

48.3.5.25 Rx FIFO 0 Configuration (M_CAN_RXF0C)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + A0h offset = A0h



M_CAN_RXF0C field descriptions

Field	Description
0 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode. 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
1–7 F0WM	Rx FIFO 0 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) >64 Watermark interrupt disabled

Table continues on the next page...

M_CAN_RXF0C field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 F0S	Rx FIFO 0 Size 0 No Rx FIFO 0 1-64 Number of Rx FIFO 0 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.26 Rx FIFO 0 Status Register (M_CAN_RXF0S)

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						RF0L	F0F	0		F0PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF0S field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. **NOTE:** Overwriting the oldest message when RXF0C.F0OM = ‘1’ will not set this flag.

Table continues on the next page...

M_CAN_RXF0S field descriptions (continued)

Field	Description
	0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

48.3.5.27 Rx FIFO 0 Acknowledge Register (M_CAN_RXF0A)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										F0AI					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF0A field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F0AI	Rx FIFO 0 Acknowledge Index. After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

48.3.5.28 Rx Buffer Configuration Register (M_CAN_RXBC)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RBSA														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXBC field descriptions

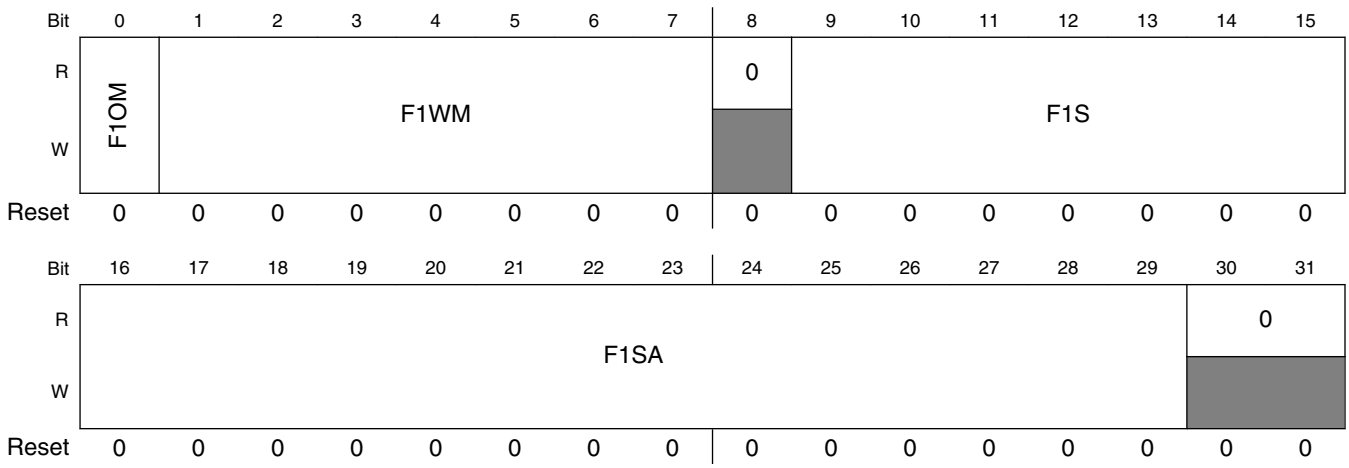
Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C. **NOTE:** Protected write (P) bit(s) which means that the write access to this field is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.29 Rx FIFO 1 Configuration Register (M_CAN_RXF1C)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + B0h offset = B0h



M_CAN_RXF1C field descriptions

Field	Description
0 F1OM	FIFO 1 Operation Mode FIFO 1 can be operated in blocking or in overwrite mode. **NOTE:** Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'. 0 FIFO 1 blocking mode 1 FIFO 1 overwrite mode
1–7 F1WM	Rx FIFO 1 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 1 watermark interrupt IR[RF1W] >64 Watermark interrupt disabled **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 F1S	**F1S** [6:0]. Rx FIFO 1 Size 0 No Rx FIFO 1

Table continues on the next page...

M_CAN_RXF1C field descriptions (continued)

Field	Description
	1-64 Number of Rx FIFO 1 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.30 Rx FIFO 1 Status Register (M_CAN_RXF1S)

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0				RF1L	F1F	0		F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		F1G1						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF1S field descriptions

Field	Description
0–1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
2–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RF1L	Rx FIFO 1 Message Lost. This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. **NOTE:** Overwriting the oldest message when RXF1C.F1OM = ‘1’ will not set this flag. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero

Table continues on the next page...

M_CAN_RXF1S field descriptions (continued)

Field	Description
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
16–17 Reserved	This field is reserved.
18–23 F1G1	Rx FIFO 1 Get Index. Rx FIFO 1 read index pointer, range 0 to 63.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 F1FL	Rx FIFO 1 Fill Level. Number of elements stored in Rx FIFO 1, range 0 to 64.

48.3.5.31 Rx FIFO 1 Acknowledge Register (M_CAN_RXF1A)

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																F1AI															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF1A field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F1AI	Rx FIFO 1 Acknowledge Index. After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

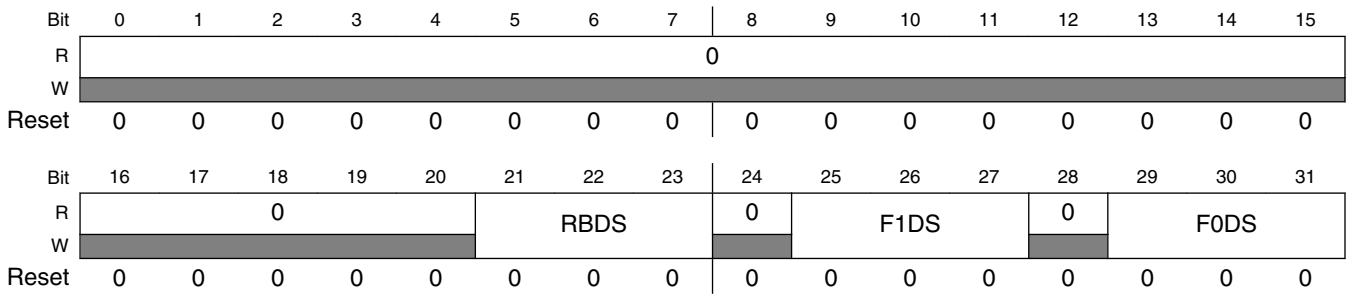
48.3.5.32 Rx Buffer / FIFO Element Size Configuration Register (M_CAN_RXESC)

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + BCh offset = BCh



M_CAN_RXESC field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 RBDS	Rx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 F1DS	Rx FIFO 1 Data Field Size **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Table continues on the next page...

M_CAN_RXESC field descriptions (continued)

Field	Description
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 F0DS	Rx FIFO 0 Data Field Size **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

48.3.5.33 Tx Buffer Configuration Register (M_CAN_TXBC)

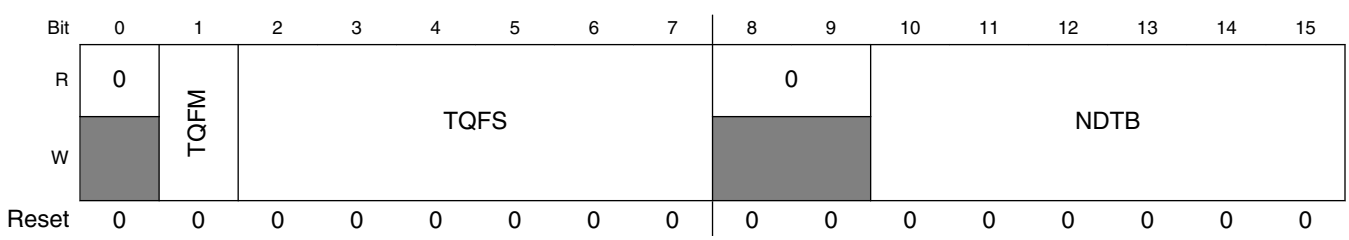
NOTE

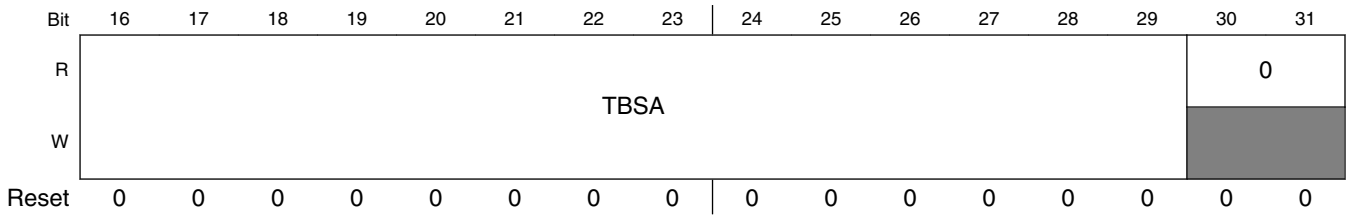
Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + C0h offset = C0h





M_CAN_TXBC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TQFM	Tx FIFO/Queue Mode **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 0 Tx FIFO operation 1 Tx Queue operation
2–7 TQFS	Tx FIFO/Queue Size **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. 0 No Tx FIFO/Queue 1-32 Number of Tx Buffers used for Tx FIFO/Queue >32 Values greater than 32 are interpreted as 32
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 NDTB	Number of Dedicated Transmit Buffers 0 No Dedicated Tx Buffers 1-32 Number of Dedicated Tx Buffers >32 Values greater than 32 are interpreted as 32 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.34 Tx FIFO/Queue Status Register (M_CAN_TXFQS)

NOTE

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			TFGI					0		TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXFQS field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11–15 TFQPI	Tx FIFO/Queue Put Index. Tx FIFO/Queue write index pointer, range 0 to 31.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = '1').
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFFL	Tx FIFO Free Level. Number of consecutive free Tx FIFO elements , range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = '1')

48.3.5.35 Tx Buffer Element Size Configuration (M_CAN_TXESC)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + C8h offset = C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													TBDS		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXESC field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TBDS	Tx Buffer Data Field Size **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'. **NOTE:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as '0xCC' (padding bytes). 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

48.3.5.36 Tx Buffer Request Pending Register (M_CAN_TXBRP)

NOTE

TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBRP field descriptions

Field	Description
0–31 TRP	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (see Tx handling) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signalled via TXBCF • after successful transmission together with the corresponding TXBTO bit • when the transmission has not yet been started at the point of cancellation • when the transmission has been aborted due to lost arbitration • when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

48.3.5.37 Tx Buffer Add Request register (M_CAN_TXBAR)

NOTE

If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBAR field descriptions

Field	Description
0–31 AR	Add Request Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

48.3.5.38 Tx Buffer Cancellation Request register (M_CAN_TXBCR)

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBCR field descriptions

Field	Description
0–31 CR	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

48.3.5.39 Tx Buffer Transmission Occurred register (M_CAN_TXBTO)

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBTO field descriptions

Field	Description
0–31 TO	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmission occurred 1Transmission occurred

48.3.5.40 Tx Buffer Cancellation Finished register (M_CAN_TXBCF)

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CF																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBCF field descriptions

Field	Description
0–31 CF	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

48.3.5.41 Tx Buffer Transmission Interrupt Enable register (M_CAN_TXBTIE)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBTIE field descriptions

Field	Description
0–31 TIE	Transmission Interrupt Enable. Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

48.3.5.42 Tx Buffer Cancellation Finished Interrupt Enable register (M_CAN_TXBCIE)

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CFIE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBCIE field descriptions

Field	Description
0–31 CFIE	Cancellation Finished Interrupt Enable. Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

48.3.5.43 Tx Event FIFO Configuration Register (M_CAN_TXEFC)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to '1'.

Address: 0h base + F0h offset = F0h

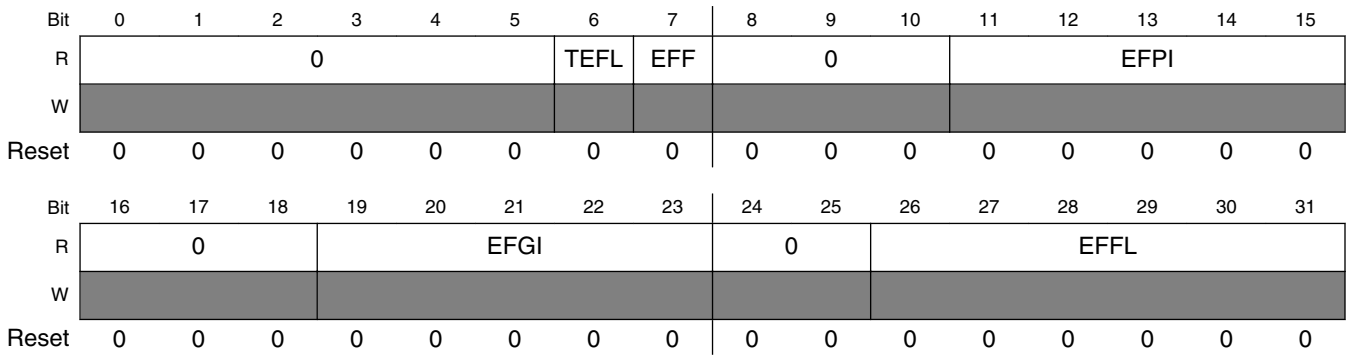
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		EFWM						0		EFS						EFSA														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXEFC field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 EFWM	Event FIFO Watermark. 0 Watermark interrupt disabled 1-32 Level for Tx Event FIFO watermark interrupt (IR[TEFW]) >32 Watermark interrupt disabled **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 EFS	Event FIFO Size. 0 Tx Event FIFO disabled 1-32 Number of Tx Event FIFO elements >32 Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
16–29 EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address, Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to '1'.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.3.5.44 Tx Event FIFO Status register (M_CAN_TXEFS)

Address: 0h base + F4h offset = F4h



M_CAN_TXEFS field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 TEFL	Tx Event FIFO Element Lost. This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full. 0 Tx Event FIFO not full 1 Tx Event FIFO full
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 EFPI	Event FIFO Put Index. Tx Event FIFO write index pointer, range 0 to 31.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 EFGI	Event FIFO Get Index. Tx Event FIFO read index pointer, range 0 to 31.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 EFFL	Event FIFO Fill Level. Number of elements stored in Tx Event FIFO, range 0 to 32.

48.3.5.45 Tx Event FIFO Acknowledge register (M_CAN_TXEFA)

Address: 0h base + F8h offset = F8h

Bit	0–26	27–31
R	0	EFAI
W		
Reset	0	0

M_CAN_TXEFA field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 EFAI	Event FIFO Acknowledge Index. After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

48.3.6 Message RAM

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM. It is not necessary to configure each of the sections listed in the following figure, nor is there any restriction with respect to the sequence of the sections.

When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

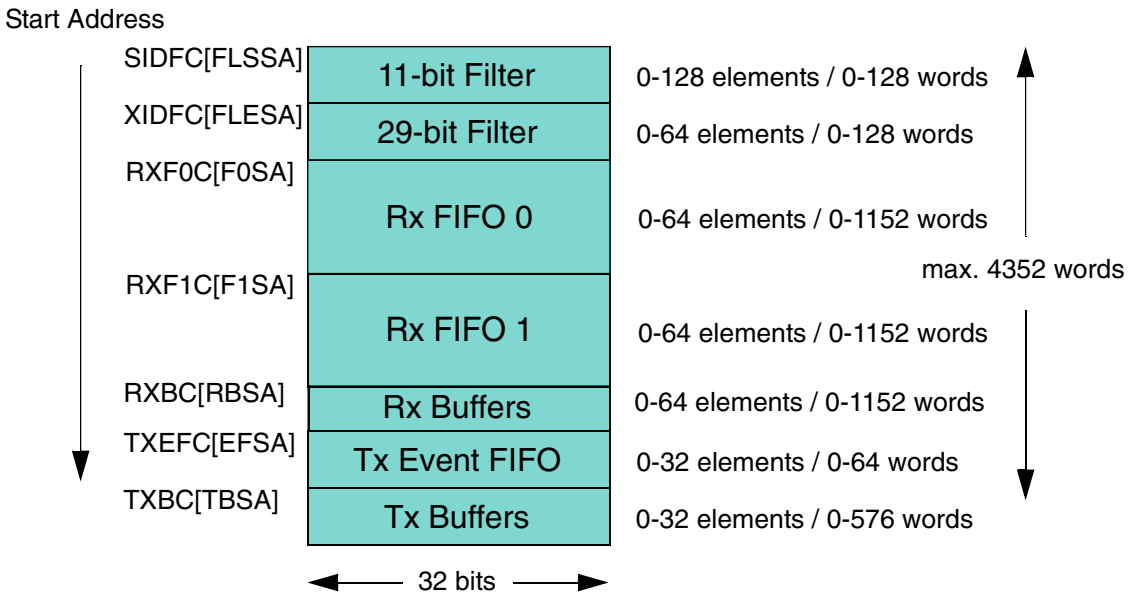


Figure 45. Message RAM Configuration

Figure 48-3. Message RAM Configuration

NOTE

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC[F0DS], RXESC[F1DS], RXESC[RBDS], and TXESC[TBDS].

The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

48.3.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

Table 48-2. Rx FIFO Element

0	7	8	15	16	23	24	31

Table continues on the next page...

Table 48-2. Rx FIFO Element (continued)

R0	ESI	XTD	RTR	ID[28:0]												
R1	ANMF	FIDX[6:0]						res	EDL	BRS	DLC[3:0]			RXTS[15:0]		
R2	DB3[7:0]							DB2[7:0]						DB1[7:0]		DB0[7:0]
R3	DB7[7:0]							DB6[7:0]						DB5[7:0]		DB4[7:0]
	...							...						...		...
Rn	DBm[7:0]							DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]

Table 48-3. Rx FIFO Element descriptions

R0 Bit 0 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 Bit 1 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier 0 11-bit standard identifier 1 29-bit extended identifier
R0 Bit 2 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame **NOTE:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = '1'), bit RTR reflects the state of the reserved bit r1.
R0 Bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 0 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE] 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 Bits 1:7 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 Bit 10 EDL Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 Bit 11 BRS Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 Bits 12:15 DLC[3:0]: Data Length Code	0-8 CAN + CAN FD: Received frame has 0-8 data bytes 9-15 CAN: Received frame has 8 data bytes 9-15 CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 16:31 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 0:7	DB3[7:0]: Data Byte 3
R2 Bits 8:15	DB2[7:0]: Data Byte 2

Table continues on the next page...

Table 48-3. Rx FIFO Element descriptions (continued)

R2 Bits 16:23	DB1[7:0]: Data Byte 1
R2 Bits 24:31	DB0[7:0]: Data Byte 0
R3 Bits 0:7	DB7[7:0]: Data Byte 7
R3 Bits 8:15	DB6[7:0]: Data Byte 6
R3 Bits 16:23	DB5[7:0]: Data Byte 5
R3 Bits 24:31	DB4[7:0]: Data Byte 4
...	...
Rn Bits 0:7	DBm[7:0]: Data Byte m
Rn Bits 8:15	DBm-1[7:0]: Data Byte m-1
Rn Bits 16:23	DBm-2[7:0]: Data Byte m-2
Rn Bits 24:31	DBm-3[7:0]: Data Byte m-3

NOTE

Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3 ..17) are used for storage of a CAN message's data field.

48.3.6.2 Tx buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC[TFQS] and TXBC[NDTB]. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

Table 48-4. Tx Buffer Element

	0	7	8	15	16	23	24	31
T0	res	XTD RTR ID[28:0]						
T1	MM[7:0]		EFC	res DLC[3:0]	res			
T2	DB3[7:0]		DB2[7:0]		DB1[7:0]		DB0[7:0]	
T3	DB7[7:0]		DB6[7:0]		DB5[7:0]		DB4[7:0]	
...	...		...		...		...	
Tn	DBm[7:0]		DBm-1[7:0]		DBm-2[7:0]		DBm-3[7:0]	

Table 48-5. Tx Buffer Element description

T0 Bit 1 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 Bit 2 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame **NOTE:** When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR.CME enables the transmission in CAN FD format.
T0 Bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 0:7 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 8 EFC	Event FIFO Control 0 Don't store Tx events 1 Store Tx events
T1 Bits 12:15 DLC[3:0]	Data Length Code 0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes 9-15 CAN FD: transmit frame has 2/16/20/24/32/48/64 data bytes
T2 Bits 0:7	DB3[7:0]: Data Byte 3
T2 Bits 8:15	DB2[7:0]: Data Byte 2
T2 Bits 16:23	DB1[7:0]: Data Byte 1
T2 Bits 24:31	DB0[7:0]: Data Byte 0
T3 Bits 0:7	DB7[7:0]: Data Byte 7
T3 Bits 8:15	DB6[7:0]: Data Byte 6
T3 Bits 16:23	DB5[7:0]: Data Byte 5
T3 Bits 24:31	DB4[7:0]: Data Byte 4
....	....
Tn Bits 0:7	DBm[7:0]: Data Byte m
Tn Bits 8:15	DBm-1[7:0]: Data Byte m-1
Tn Bits 16:23	DBm-2[7:0]: Data Byte m-2
Tn Bits 24:31	DBm-3[7:0]: Data Byte m-3

NOTE

Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message's data field.

48.3.6.3 Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

Table 48-6. Tx Event FIFO Element

	0	7	8	15	16	23	24	31
E0	ESI	XTD RTR	ID[28:0]					
E1	MM[7:0]		ET[1:0]	EDL BRS	DLC[3:0]	TXTS[15:0]		

Table 48-7. Tx Event FIFO Element

E0 Bit 0 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 Bit 1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 Bit 2 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 Bits 3:31 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 Bits 0:7 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status
E1 Bits 8:9ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 Bit 10 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 Bit 11 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 Bits 12:15 DLC[3:0]	Data Length Code 0-8 Frame with 0-8 data bytes transmitted

Table continues on the next page...

Table 48-7. Tx Event FIFO Element (continued)

	9-15 Frame with 8 data bytes transmitted 9-15 CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1 Bits 16:31TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

48.3.6.4 Standard message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC.FLSSA plus the index of the filter element (0…127).

Table 48-8. Standard Message ID Filter Element

	0		7	8	15	16	23	24	31
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		res	SFID2[10:0]		

Table 48-9. Standard Message ID Filter Element Field Description

Bits 0:1 SFT[1:0]	Standard Filter Type First ID of standard ID filter element. 00 Range filter from SF1ID to SF2ID (SF2ID > SF1ID) 01 Dual ID filter for SF1ID or SF2ID 10 Classic filter: SF1ID = filter, SF2ID = mask 11 Reserved
Bit 2:4 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag IE[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches

Table continues on the next page...

Table 48-9. Standard Message ID Filter Element Field Description (continued)

		100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of SFT[1:0] ignored
Bits 5:15 SFID1[10:0]		Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a tandard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 21:31	SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: SFEC = '001'...'110' Second ID of standard ID filter element SFEC = '111' Filter for Rx Buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	SFID2[8:6]	Is used to control the M_CAN filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN host clock period in case the filter matches.
	SFID2[5:0]	Defines the offset to the Rx Buffer Start Address RXBC[RBSA] for storage of a matching message.

48.3.6.5 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0…63).

Table 48-10. Extended Message ID Filter Element

0		7	8	15	16	23	24	31
F0	EFEC[2:0]	EFID1[28:0]						

Table continues on the next page...

Table 48-10. Extended Message ID Filter Element (continued)

F1	EFT[1:0]	res	EFID2[28:0]

Table 48-11. Extended Message ID Filter Element Field Description

F0 Bits 0:2 EFEC[2:0]		Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 Bits 3:31 **EFID1[28:0]**		Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffer or debug messages this field defines the ID of a extended debug message to be stored. The received identifiers must match exactly, no masking mechanism is used.
F1 Bits 0:1 EFT[1:0]		Extended Filter Type 00 — Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) 01 — Dual ID filter for EF1ID or EF2ID 10 — Classic filter: EF1ID = filter, EF2ID = mask 11 — Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bits 3:31	EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = “001”...”110” Second ID of extended ID filter element

Table continues on the next page...

Table 48-11. Extended Message ID Filter Element Field Description (continued)

		2) EFEC = “111” Filter for Rx Buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00= Store message into an Rx Buffer 01= Debug Message A 10= Debug Message B 11= Debug Message C
	EFID2[8:6]	It is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN clock period in case the filter matches
	EFID2[5:0]	Defines the offset to the Rx Buffer Start Address RXBC[RBSA] for storage of a matching message.

48.3.7 M_CAN functional description

48.3.7.1 Operating modes

48.3.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = '1'. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS–High Priority Message Status
- RXF0S–Rx FIFO 0 Status
- RXF1S–Rx FIFO 1 Status
- TXFQS–Tx FIFO/Queue Status
- TXBRP–Tx Buffer Request Pending
- TXBTO–Tx Buffer Transmission Occurred
- TXBCF–Tx Buffer Cancellation Finished
- TXEFS–Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = '1'.

The following registers are only writable while CCCR[CCE] = '0'

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR[TEST] and CCCR.MON can only be set by the Host while CCCR[INIT] = '1' and CCCR[CCE] = '1'. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = '1' and CCCR[CCE] = '1'.

Following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
          while (read_data != 0x003){
            write(MCAN0,0xF,driver.regs.can0_cccr.addr,0x003);
    read_data = read(MCAN0,0xF,driver.regs.can0_cccr.addr);
          };

-- Configuring the time stamp and timeout registers
          write(MCAN0,0xF,driver.regs.can0_tscc.addr,0x000);
          write(MCAN0,0xF,driver.regs.can0_tocc.addr,0x000);

-- Configuring the interrupt registers
          write(MCAN0,0xF,driver.regs.can0_ie.addr,0xFFFFFFFF);
          write(MCAN0,0xF,driver.regs.can0_ils.addr,0x00000002);
          write(MCAN0,0xF,driver.regs.can0_ile.addr,0x00000003);

          write(MCAN0,0xF,driver.regs.can0_gfc.addr,0x000);
          write(MCAN0,0xF,driver.regs.can0_xidam.addr,0x00FFFFFF);

--Writing the start addresses of all the MCAN Buffers
          generate and drive MCAN0 can0_rxf0c keeping {
                                                        .f0sa == 0x100;
```

```
                                                        .f0s  == 0x03;
                                                        .f0wm == 0x01;
                                                        };
          generate and drive MCAN0 can0_rxf1c keeping {
                                                        .f1sa == 0x200;
                                                        .f1s  == 0x03;
                                                        .f1wm == 0x01;
                                                        };
          generate and drive MCAN0 can0_sidfc keeping {
                                                        .flssa == 0x000;
                                                        .lss   == 0x80;
                                                        };
          generate and drive MCAN0 can0_xidfc keeping {
                                                        .flesa == 0x080;
                                                        .lse   == 0x40;
                                                        };
          generate and drive MCAN0 can0_txbc  keeping {
                                                        .tbsa == 0x300;
                                                        .ndtb == 0x020;
                                                        .tfqs == 0x1D;
                                                        };
          generate and drive MCAN0 can0_txefc keeping {
                                                        .efsa == 0x380;
                                                        .efs == 0x020;
                                                        .efwm == 0x00;
                                                        };
--Writing to the RAM for filter configuration
================================================
--   Writing to TX Buffer
          write(RAM,0xF,0xC00,0x04040000);
          write(RAM,0xF,0xC04,0x10830000);
          write(RAM,0xF,0xC08,0x00010001);
          write(RAM,0xF,0xC10,0x50000002);
          write(RAM,0xF,0xC14,0x11830000);
          write(RAM,0xF,0xC18,0x00070102);

--   Writing to s-filter
          write(RAM,0xF,0x000,0x58020004);
          write(RAM,0xF,0x004,0x08000005);
          write(RAM,0xF,0x008,0x58060007);
          write(RAM,0xF,0x00C,0x08030008);
          write(RAM,0xF,0x010,0x880E07FB);
          write(RAM,0xF,0x014,0x980D07FE);
          write(RAM,0xF,0x018,0x68100100);
          write(RAM,0xF,0x01C,0x70120200);
          write(RAM,0xF,0x020,0x981F07F6);
          write(RAM,0xF,0x024,0x50140014);
          write(RAM,0xF,0x028,0x58160017);
          write(RAM,0xF,0x02C,0x08110019);
          write(RAM,0xF,0x030,0x4A22001A);
          write(RAM,0xF,0x034,0xA01A07FF);
          write(RAM,0xF,0x038,0x5BFF07E3);
          write(RAM,0xF,0x03C,0x080007FF);

--   Writing to x-filter
          write(RAM,0xF,0x200,0x201ABCD9);
          write(RAM,0xF,0x204,0x001ABCE5);
          write(RAM,0xF,0x208,0x403BCDEF);
          write(RAM,0xF,0x20C,0x003BCDEF);
          write(RAM,0xF,0x210,0x605CDEF2);
          write(RAM,0xF,0x214,0x405CDEF0);
          write(RAM,0xF,0x218,0x407DEEFF);
          write(RAM,0xF,0x21C,0x007DEF01);
          write(RAM,0xF,0x220,0x209ABCDE);
          write(RAM,0xF,0x224,0x009ABCDF);
          write(RAM,0xF,0x228,0xA0BBCDEF);
          write(RAM,0xF,0x22C,0x40BBCDEE);
          write(RAM,0xF,0x230,0x20DCDEF0);
          write(RAM,0xF,0x234,0x9FFFFFFF);
          write(RAM,0xF,0x238,0x40000000);
```

```
          write(RAM,0xF,0x23C,0x1FFFFFFF);
================================================
write(MCAN0,0xF,driver.regs.can0_btp.addr,0x00002303);

          print "Configuration done";
          read_data = 0x01;
          while (read_data != 0x00){
            write(MCAN0,0xF,driver.regs.can0_cccr.addr, 0x000);
    wait [1];
    read_data = read(MCAN0,0xF,driver.regs.can0_cccr.addr);
          };

          print "Initialization done";
```

48.3.7.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

48.3.7.3 CAN FD Operations

There are two stages in the CAN FD protocol, first the Long Frame Mode where the data field of a CAN frame may be longer that 8 bytes. The second stage is the Fast Frame Mode where control field, data field, and CRC field of a CAN Frame are transmitted with a higher bit rate than the beginning and the end of the frame. Fast Frame Mode can only be used in combination with Long Frame Mode.

The CAN operation mode is chosen by programming CCCR.CME. In case CCCR.CME = “01” transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CCCR.CME = “10”/“11” transmission and reception of long and fast CAN FD frames is enabled. CCCR.CME can only be changed while CCCR.INIT and CCCR.CCE are both set.

When initialization is left (CCCR.INIT set to ‘0’), both Long Frame Mode and Fast Frame Mode are inactive, they have to be requested by writing to CCCR.CMR.

A mode change requested by writing to CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event CCCR.CMR is reset to “00” and the status flags CCCR.FDBS and CCCR.FDO are set

accordingly. In case the requested CAN operation mode is not enabled, the value written to CCCR.CMR is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1.

It is not necessary to change the CAN operation mode after system startup. A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting according to ISO11898-1 until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to CAN communication according ISO11898-1.

When CCCR.CME is not ‘00’, received CAN FD frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD frame, EDL = dominant signifies a standard CAN frame. In a CAN FD frame, the two bits following EDL, r0 and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

Reception of CAN frames according to ISO 11898-1 is possible in all CAN operation modes.

The status bits CCCR.FDO and CCCR.FDBS indicate the format of transmitted frames. When CCCR.FDO is set, frames will be transmitted in CAN FD format with EDL = recessive. When both CCCR.FDO and CCCR.FDBS are set, frames will be transmitted in CAN FD format with bit rate switching and both bits EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table.

Table 48-12. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing & Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing & Prescaler Register FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

48.3.7.3.1 Transceiver Delay Compensation

The length of the bus line has no impact. When transmitting via M_CAN_Tx pin the protocol controller receives the transmitted data from its local CAN transceiver via M_CAN_Rx pin. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. The following figure below describes how the transceiver loop delay is measured.

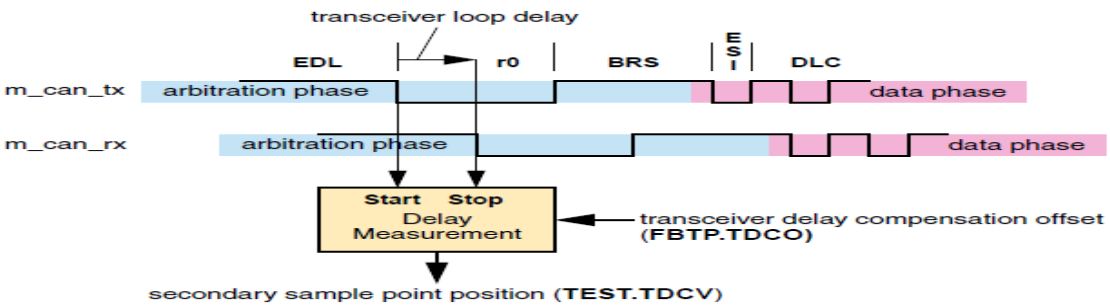


Figure 48-4. Transceiver delay measurement

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin M_CAN_Tx and the data received at M_CAN_Rx pin, starting with the falling edge of bit EDL. The delay is measured in M_CAN core clock periods.

A secondary sample point is calculated by adding a configurable transceiver delay compensation offset (FBTP.TDCO) to the measured transceiver delay. The transceiver delay compensation offset is used to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta tq.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following sample point. During arbitration phase the delay compensation is always disabled.

For the transceiver delay compensation the following boundary conditions have to be considered:

- The sum of the measured delay from M_CAN_Tx to M_CAN_Rx and the configured transceiver delay compensation offset FBTP.TDCO has to be less than 3 bit times in the data phase.
- The sum of the measured delay from M_CAN_Tx to M_CAN_Rx and the configured transceiver delay compensation offset FBTP.TDCO has to be less or equal 63 M_CAN core clock periods. In case this sum exceeds 63 M_CAN core clock periods, the maximum value of 63 M_CAN core clock periods is used for transceiver delay compensation.

The actual delay compensation value is monitored by reading TEST.TDCV

48.3.7.3.2 Configuration and status

Compensation for the transceiver loop delay by the M_CAN is enabled via FBTP[TDC]. The transceiver delay compensation offset is configured via FBTP[TDCO]. The actual delay compensation value applied by the M_CAN's protocol engine can be read from TEST[TDCV].

48.3.7.4 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCR[ASM]. The bit can only be set by the Host when both CCCR[CCE] and CCCR[INIT] are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset CCCR[ASM].

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

48.3.7.5 Bus monitoring mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR[MON] to one. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

The following figure shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus Monitoring Mode.

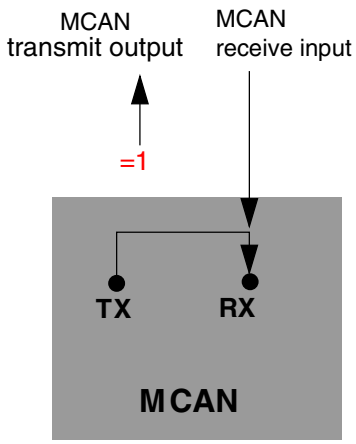


Figure 48-5. Pin Control in Bus Monitoring Mode

48.3.7.6 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

48.3.7.6.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:

- Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
- Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

48.3.7.7 Power down (Sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one. When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs (CAN clock and host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

48.3.7.8 Test modes

To enable write access to register TEST (see Test Register (M_CAN_TEST)), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_can_tx by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at MCAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

Note

Test modes should be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

48.3.7.8.1 External Loop Back mode

The M_CAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if the y pass acceptance filtering) into Rx FIFOs. The following figure shows the connection of MCAN Tx and Rx signals in External Loop Back mode. This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the MCAN Tx pin.

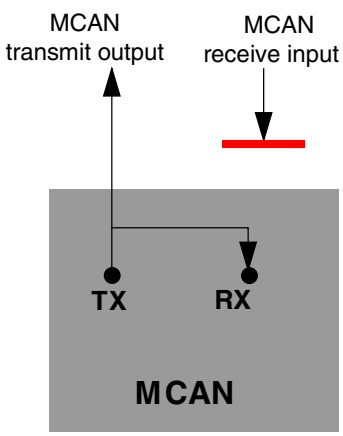


Figure 48-6. External Loop Back mode

48.3.7.8.2 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a "Hot Selftest", meaning the M_CAN can be tested without affecting a running CAN system connected to the MCAN Tx and Rx pins. In this mode MCAN Tx pin is disconnected from the M_CAN and MCAN Tx pin is held recessive. The following figure shows the connection of MCAN Tx pin and Rx to the M_CAN in case of Internal Loop Back mode.

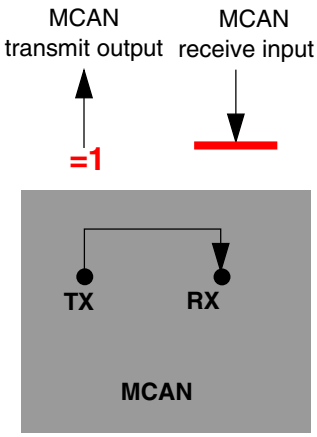


Figure 48-7. Internal loop back mode

48.3.8 Timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of a Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

48.3.9 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = '0'. It is stopped when CCCR[INIT] = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

48.3.10 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFOs Put and Get Indices.

48.3.10.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR[LEC] respectively PSR[FLEC].

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR[LEC] respectively PSR[FLEC]. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in "Rx FIFO Overwrite Mode" section have to be considered.

Note

When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

48.3.10.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID. There are two possibilities when range filtering is used together with extended frames:

- EFT = "00": The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = "11": The Extended ID AND Mask (XIDAM) is not used for range filtering

48.3.10.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

48.3.10.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

48.3.10.1.4 Standard message ID filtering

Table 48-8 shows the flow for standard message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in Standard message ID Filter element.

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

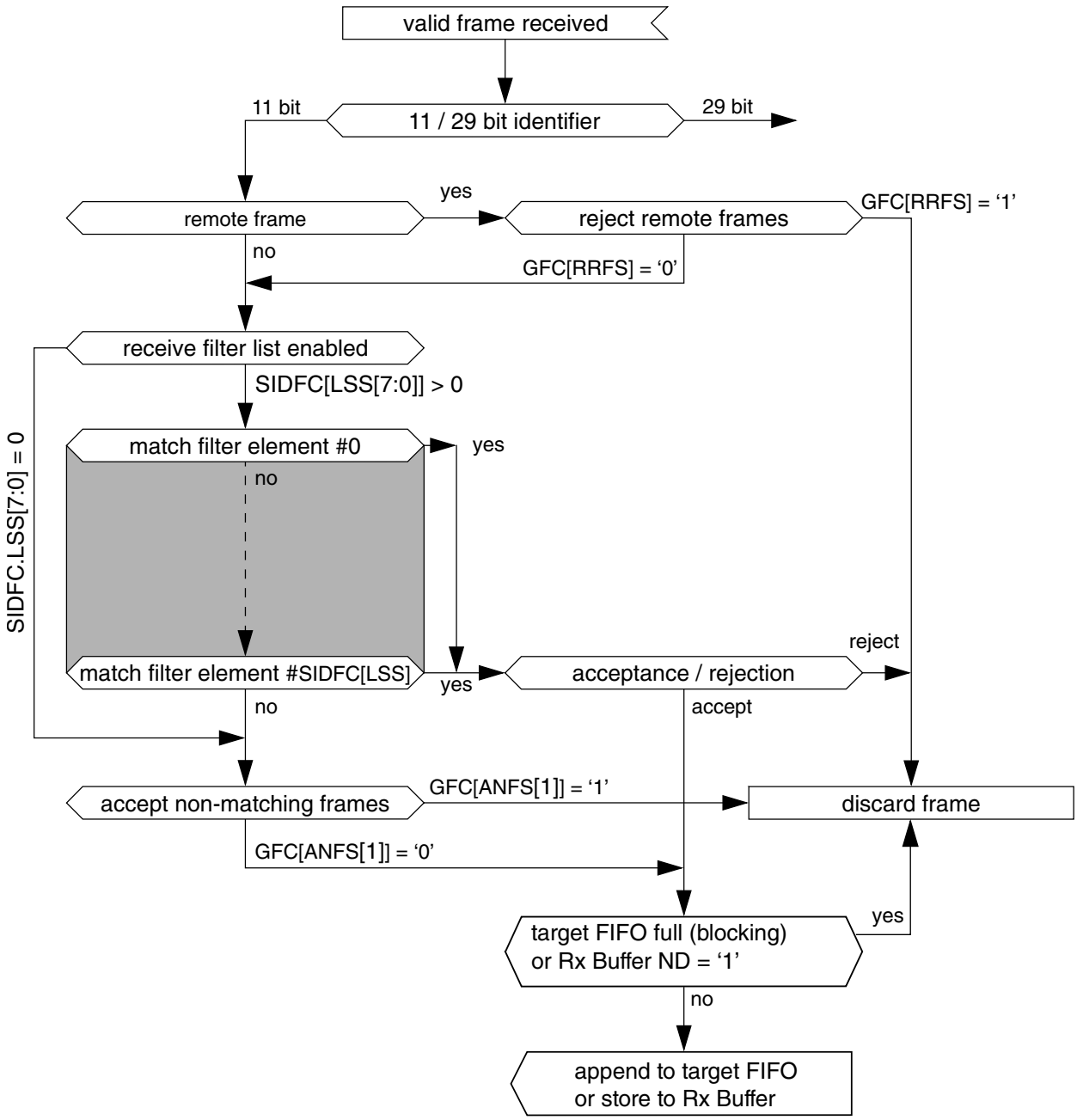


Figure 48-8. Standard Message ID Filter Path

48.3.10.1.5 Extended message ID filtering

The figure below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in the Extended ID Filter Configuration Register (M_CAN_XIDFC).

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

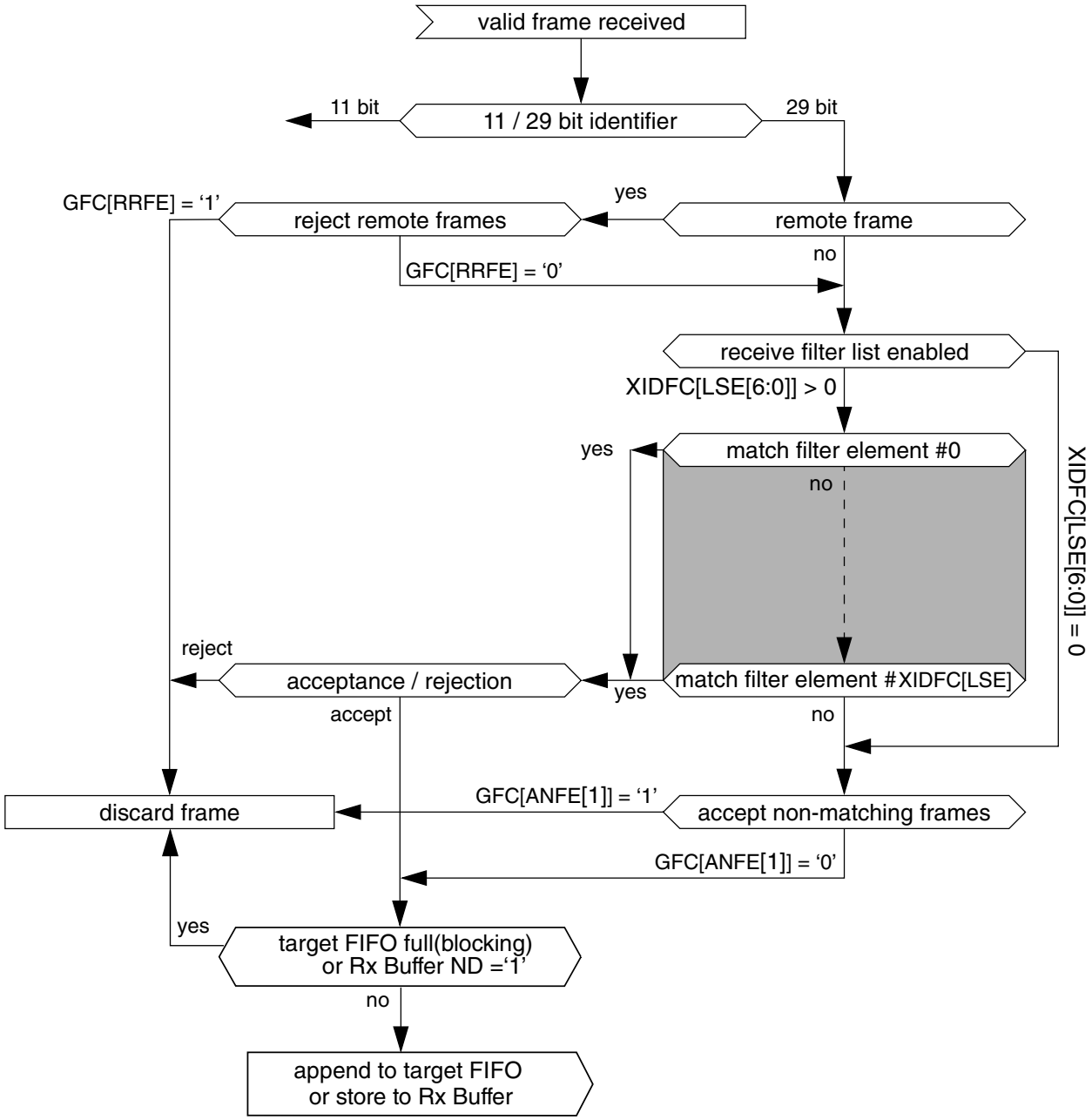


Figure 48-9. Extended Message ID Filter Path

48.3.10.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C. Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see Acceptance filtering. The Rx FIFO element is described in Rx Buffer and FIFO element.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC[FnWM], interrupt flag IR[RFnW] is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS[FnF]. In addition interrupt flag IR[RFnF] is set.

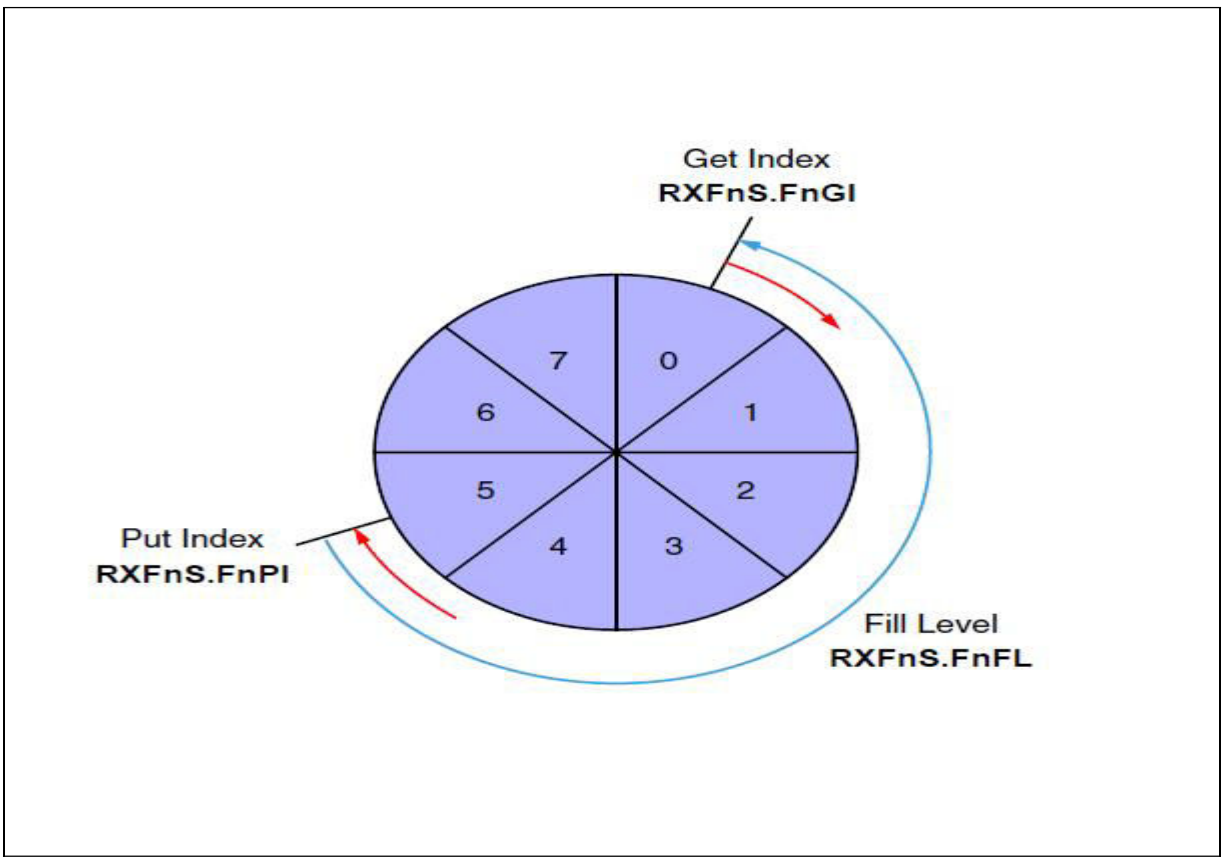


Figure 48-10. Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS[FnGI] x FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

Table 48-13. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

48.3.10.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC[FnOM] = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS[FnPI] = RXFnS[FnGI]), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS[FnF] = ‘1’. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS[RFnL] = ‘1’. In addition interrupt flag IR[RFnL] is set.

48.3.10.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC[FnOM]= ‘1’.

When an Rx FIFO full condition (RXFnS[FnPI] = RXFnS[FnGI]) is signalled by RXFnS[FnF] = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. Figure 9 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

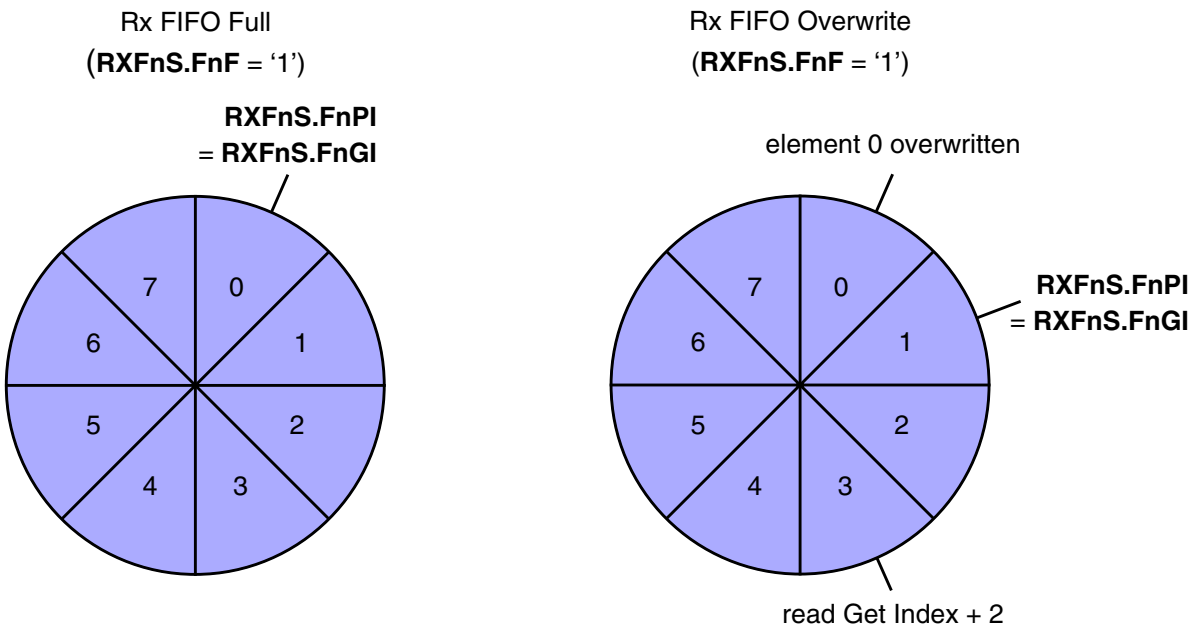


Figure 48-11. Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA[FnA]. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS[FnF] = '0').

48.3.11 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = "111" and SFID2 / EFID2[10:9] = "00" has to be configured.

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 48-14. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration

48.3.11.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

48.3.12 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element.

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = “111” have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_can_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while m_can_dma_req is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets m_can_dma_ack. This resets m_can_dma_req. Now the M_CAN is prepared to receive the next set of debug messages.

NOTE

To use full ‘Debug on CAN Support’ feature on a M_CAN instance, a DMA channel is required. Refer to device DMA map, to see which M_CAN instances has been assigned with DMA channel.

48.3.12.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to “111”. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR[DRX] are set. The reception of debug messages can be monitored via RXF1S[DMS].

Table 48-15. Example Filter Configuration for Debug Messages

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

48.3.12.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S[DMS].

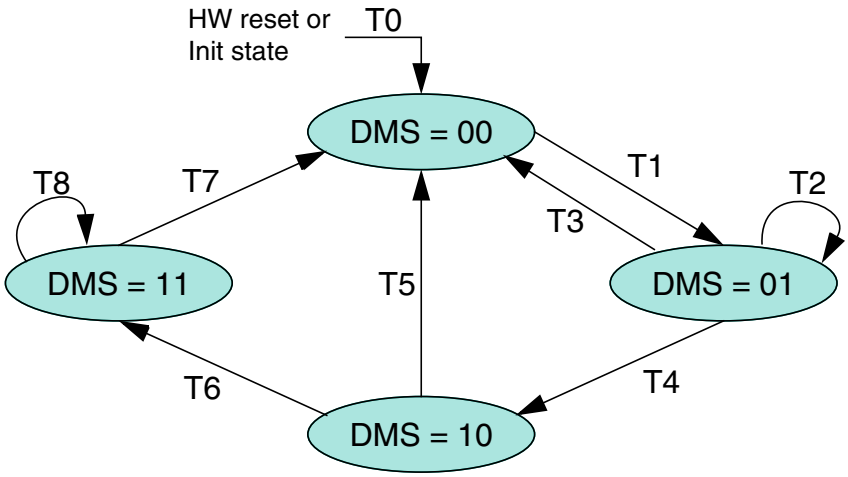


Figure 48-12. Debug Message Handling State Machine

T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C

T1: reception of debug message A

T2: reception of debug message A

T3: reception of debug message C

T4: reception of debug message B

T5: reception of debug messages A, B

T6: reception of debug message C

T7: DMA transfer completed

T8: reception of debug message A,B,C (message rejected)

48.3.13 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message handling state machine enters idle state (DMS = "00") and waits for reception of the next debug messages.

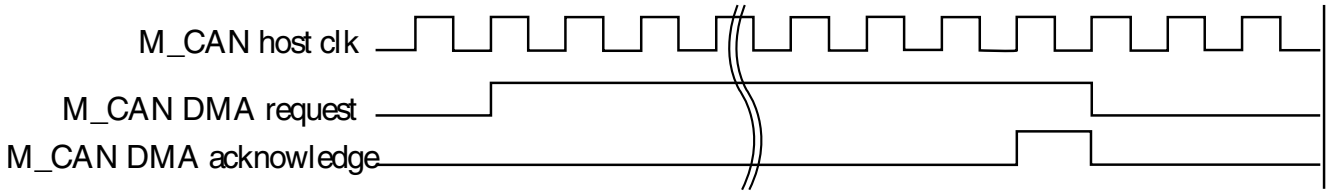


Figure 48-13. Timing of DMA interface signals

48.3.14 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in Tx buffer element.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

48.3.14.1 Transmit Pause

The Rx FIFO overwrite mode is configured by RXFnC[FnOM]= ‘1’.

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU’s CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR[TXP]. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR[TXP] = ‘0’).

This feature looses up burst transmissions coming from a single node and it protects against “babbling idiot” scenarios where the application program erroneously requests too many transmissions.

48.3.14.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) to the Tx Buffer Start Address TXBC[TBSA].

Table 48-16. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

48.3.14.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFOs Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) x Element Size to the Tx Buffer Start Address TXBC[TBSA].

48.3.14.4 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx FIFO/Queue Put Index TXFQS[TFQPI] (0…31) x Element Size to the Tx Buffer Start Address TXBC[TBSA].

48.3.14.5 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

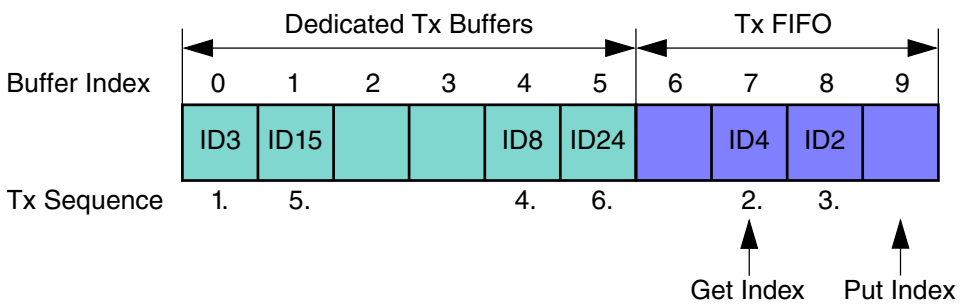


Figure 48-14. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

48.3.14.6 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

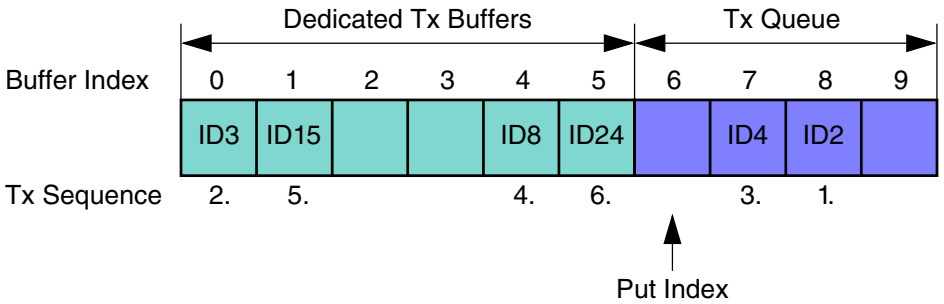


Figure 48-15. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

48.3.14.7 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

48.3.14.8 Tx Event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in Tx event FIFO element. When a Tx Event FIFO full condition is signalled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

48.3.15 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see Rx FIFO 0 Acknowledge Register (M_CAN_RXF0A), Rx FIFO 1 Acknowledge Register (M_CAN_RXF1A) and Tx Event FIFO Acknowledge register (M_CAN_TXEFA)) . Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the

two Rx FIFOs. In this case the FIFOs Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

Note

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

48.4 Time Triggered Modular CAN (M_TTCAN) core

The M_TTCAN performs communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 part A,B) and according to ISO 11898-4 (time-triggered communication on CAN). The M_TTCAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation.

In addition the M_TTCAN supports communication according to CAN FD protocol specification 1.0 with data fields of up to eight bytes length. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

The message storage is intended to be a single-ported or dual-ported Message RAM outside of the module. It is connected to the M_TTCAN via the Generic Master Interface. Depending on the chosen ASIC integration, multiple M_TTCAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_TTCAN can be connected to a wide range of Host CPUs via its 8/16/32-bit Generic Slave Interface. The M_TTCAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

48.4.1 Features

The following are the features of M_TTCAN.

- M_TTCAN protocol level 1 and level 2 completely in hardware
- AUTOSAR optimized
- SAE J1939 optimized
- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1, -4
- CAN FD with maximum 8 data bytes supported
- Event synchronized time-triggered communication supported
- CAN Error Logging
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 Transmit Buffers
- Configurable Transmit FIFO / Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_TTCAN share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8/16/32bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug on CAN support

48.4.2 Block Diagram

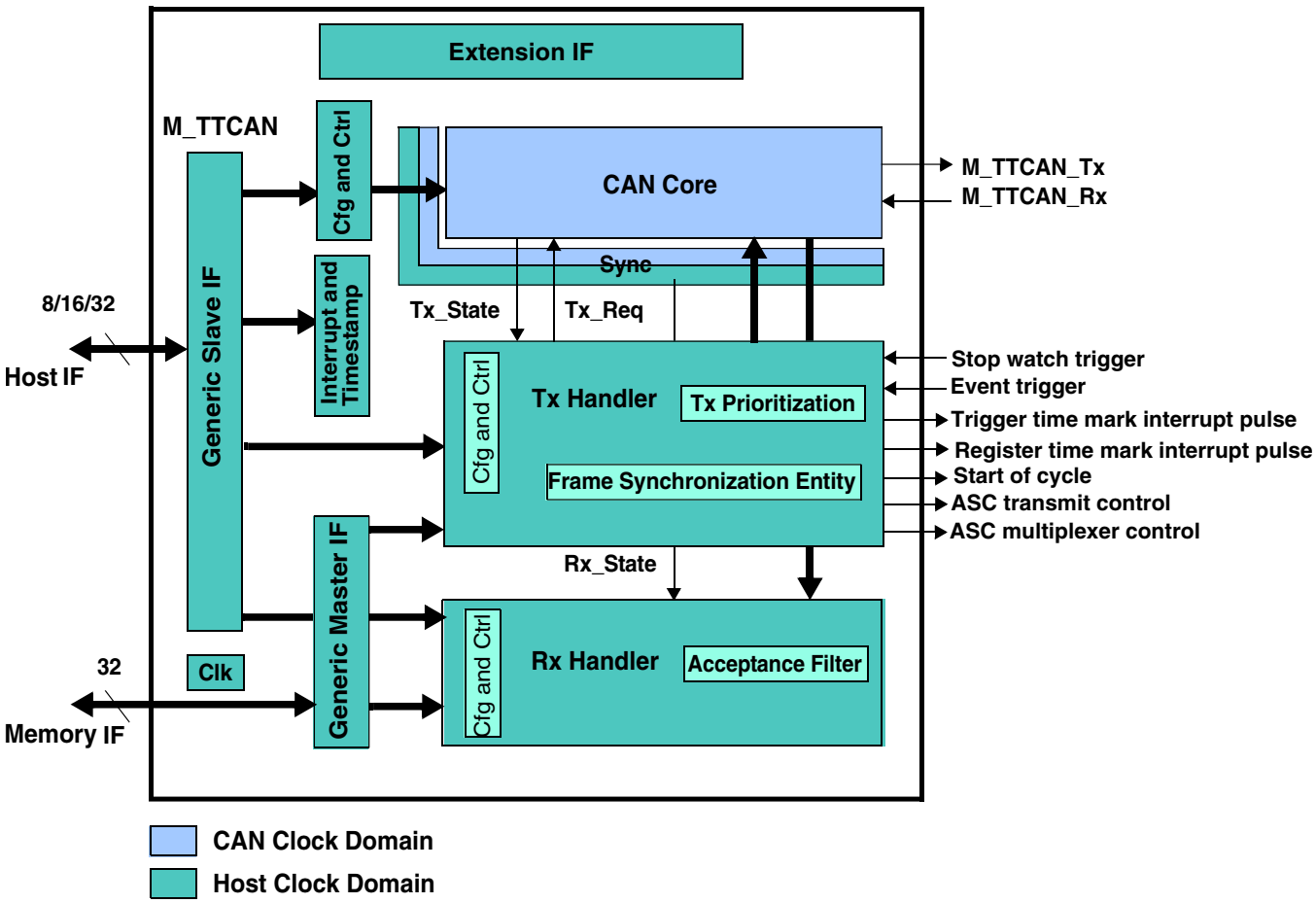


Figure 48-16. M_TTCAN Core block diagram

- CAN Core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Cfg and Ctrl: CAN Core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported. The Tx Handler

also implements the Frame Synchronization Entity FSE which controls time-triggered communication according to ISO11898-4. It synchronizes itself to the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the Message RAM. Stop Watch Trigger, Event Trigger, and Time Mark Interrupt are synchronization interfaces.

- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Generic Slave Interface: Connects the M_TTCAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_TTCAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bit. Refer Table 48-38.
- Extension Interface: All flags from the Interrupt Register IR and TT Interrupt Register TTIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_TTCAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

48.4.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as M_TTCAN clock. The CAN Core should be programmed to have at least 8 clocks per bit time, this is e.g. 1 Mbaud at M_TTCAN clock ≤ 8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN Core needs not to be higher than 8 MHz.

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note

In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

CAUTION

It is strongly recommended to stop this module when the device is in the STOP mode or in case, if the CAN message reception is mandatory, please check if the PLL is unlocked before transmitting any message, or use the external oscillator instead of the IRC during the STOP mode.

48.4.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to M_TTCAN INT 0 or to M_TTCAN INT 1. By default all interrupts are routed to interrupt line M_TTCAN INT 0. By programming EINT0 and EINT1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

48.4.5 Memory map and register description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in the memory map. Additionally the Bus_Off state is reset and the CAN transmit output is set to recessive (HIGH). The value 0x0001 (CCCR[INIT] = '1') in the CC Control Register enables software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets CCCR[INIT] to '0'.

The M_TTCAN module allocates an address space of 512 bytes. All registers are organized as 32-bit registers. The M_TTCAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

NOTE

Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only with CCCR[CCE]= '1' and CCCR[INIT]= '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

M_TTCAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Core Release Register (M_TTCAN_CREL)	32	R	See section	48.4.5.1/ 1944
4	Endian Register (M_TTCAN_ENDN)	32	R	8765_4321h	48.4.5.2/ 1944
C	Fast Bit Timing and Prescaler Register (M_TTCAN_FBTP)	32	R/W	0000_0A33h	48.4.5.3/ 1945
10	Test Register (M_TTCAN_TEST)	32	R/W	0000_0000h	48.4.5.4/ 1946
14	RAM Watchdog Register (M_TTCAN_RWD)	32	R/W	0000_0000h	48.4.5.5/ 1949
18	CC Control Register (M_TTCAN_CCCR)	32	R/W	0000_0001h	48.4.5.6/ 1950
1C	Bit Timing and Prescaler Register (M_TTCAN_BTP)	32	R/W	0000_0A33h	48.4.5.7/ 1952
20	Timestamp Counter Configuration Register (M_TTCAN_TSCC)	32	R	0000_0000h	48.4.5.8/ 1953
24	Timestamp Counter Value Register (M_TTCAN_TSCV)	32	R/W	0000_0000h	48.4.5.9/ 1954
28	Timeout Counter Configuration Register (M_TTCAN_TOCC)	32	R/W	FFFF_0000h	48.4.5.10/ 1955
2C	Timeout Counter Value Register (M_TTCAN_TOCV)	32	R/W	0000_FFFFh	48.4.5.11/ 1956
40	Error Counter Register (M_TTCAN_ECR)	32	R	0000_0000h	48.4.5.12/ 1956
44	Protocol Status Register (M_TTCAN_PSR)	32	R	0000_0707h	48.4.5.13/ 1957
50	Interrupt Register (M_TTCAN_IR)	32	R/W	0000_0000h	48.4.5.14/ 1960
54	Interrupt Enable Register (M_TTCAN_IE)	32	R/W	0000_0000h	48.4.5.15/ 1963
58	Interrupt Line Select Register (M_TTCAN_ILS)	32	R/W	0000_0000h	48.4.5.16/ 1967
5C	Interrupt Line Enable Register (M_TTCAN_ILE)	32	R/W	0000_0000h	48.4.5.17/ 1970

Table continues on the next page...

M_TTCAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	Global Filter Configuration Register (M_TTCAN_GFC)	32	R/W	0000_0000h	48.4.5.18/ 1971
84	Standard ID Filter Configuration Register (M_TTCAN_SIDFC)	32	R/W	0000_0000h	48.4.5.19/ 1972
88	Extended ID Filter Configuration Register (M_TTCAN_XIDFC)	32	R/W	0000_0000h	48.4.5.20/ 1973
90	Extended ID and Mask Register (M_TTCAN_XIDAM)	32	R/W	1FFF_FFFFh	48.4.5.21/ 1974
94	High Priority Message Status Register (M_TTCAN_HPMS)	32	R	0000_0000h	48.4.5.22/ 1974
98	New Data 1 Register (M_TTCAN_NDAT1)	32	R/W	0000_0000h	48.4.5.23/ 1975
9C	New Data 2 Register (M_TTCAN_NDAT2)	32	R/W	0000_0000h	48.4.5.24/ 1975
A0	Rx FIFO 0 Configuration Register (M_TTCAN_RXF0C)	32	R/W	0000_0000h	48.4.5.25/ 1976
A4	Rx FIFO 0 Status Register (M_TTCAN_RXF0S)	32	R	0000_0000h	48.4.5.26/ 1977
A8	Rx FIFO 0 Acknowledge Register (M_TTCAN_RXF0A)	32	R/W	0000_0000h	48.4.5.27/ 1978
AC	Rx Buffer Configuration Register (M_TTCAN_RXBC)	32	R/W	0000_0000h	48.4.5.28/ 1978
B0	Rx FIFO 1 Configuration Register (M_TTCAN_RXF1C)	32	R/W	0000_0000h	48.4.5.29/ 1979
B4	Rx FIFO 1 Status Register (M_TTCAN_RXF1S)	32	R	0000_0000h	48.4.5.30/ 1980
B8	Rx FIFO 1 Acknowledge register (M_TTCAN_RXF1A)	32	R/W	0000_0000h	48.4.5.31/ 1981
C0	Tx Buffer Configuration register (M_TTCAN_TXBC)	32	R/W	0000_0000h	48.4.5.32/ 1981
C4	Tx FIFO/Queue Status register (M_TTCAN_TXFQS)	32	R	0000_0000h	48.4.5.33/ 1982
CC	Tx Buffer Request Pending register (M_TTCAN_TXBRP)	32	R	0000_0000h	48.4.5.34/ 1984
D0	Tx Buffer Add Request register (M_TTCAN_TXBAR)	32	R/W	0000_0000h	48.4.5.35/ 1985
D4	Tx Buffer Cancellation Request register (M_TTCAN_TXBCR)	32	R/W	0000_0000h	48.4.5.36/ 1985
D8	Tx Buffer Transmission Occurred register (M_TTCAN_TXBTO)	32	R	0000_0000h	48.4.5.37/ 1986
DC	Tx Buffer Cancellation Finished register (M_TTCAN_TXBCF)	32	R	0000_0000h	48.4.5.38/ 1986
E0	Tx Buffer Transmission Interrupt Enable register (M_TTCAN_TXBTIE)	32	R/W	0000_0000h	48.4.5.39/ 1987

Table continues on the next page...

M_TTCAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E4	Tx Buffer Cancellation Finished Interrupt Enable register (M_TTCAN_TXBCIE)	32	R/W	0000_0000h	48.4.5.40/ 1987
F0	Tx Event FIFO Configuration register (M_TTCAN_TXEFC)	32	R/W	0000_0000h	48.4.5.41/ 1988
F4	Tx Event FIFO Status register (M_TTCAN_TXEFS)	32	R	0000_0000h	48.4.5.42/ 1989
F8	Tx Event FIFO Acknowledge register (M_TTCAN_TXEFA)	32	R/W	0000_0000h	48.4.5.43/ 1990
100	TT Trigger Memory Configuration register (M_TTCAN_TTTMC)	32	R/W	0000_0000h	48.4.5.44/ 1990
104	TT Reference Message Configuration register (M_TTCAN_TTRMC)	32	R/W	0000_0000h	48.4.5.45/ 1991
108	TT Operation Configuration register (M_TTCAN_TTOCF)	32	R/W	0001_0000h	48.4.5.46/ 1992
10C	TT Matrix Limits register (M_TTCAN_TTMLM)	32	R/W	0000_0000h	48.4.5.47/ 1994
110	TUR Configuration register (M_TTCAN_TURCF)	32	R/W	1000_0000h	48.4.5.48/ 1995
114	TT Operation Control register (M_TTCAN_TTOCN)	32	R/W	0000_0000h	48.4.5.49/ 1998
118	TT Global Time Preset register (M_TTCAN_TTGTP)	32	R/W	0000_0000h	48.4.5.50/ 2001
11C	TT Time Mark register (M_TTCAN_TTTMK)	32	R/W	0000_0000h	48.4.5.51/ 2001
120	TT Interrupt Register (M_TTCAN_TTIR)	32	R/W	0000_0000h	48.4.5.52/ 2003
124	TT Interrupt Enable register (M_TTCAN_TTIE)	32	R/W	0000_0000h	48.4.5.53/ 2005
128	TT Interrupt Line Select register (M_TTCAN_TTILS)	32	R/W	0000_0000h	48.4.5.54/ 2007
12C	TT Operation Status register (M_TTCAN_TTOST)	32	R	0000_0080h	48.4.5.55/ 2010
130	TUR Numerator Actual register (M_TTCAN_TURNA)	32	R	0001_0000h	48.4.5.56/ 2012
134	TT Local and Global Time register (M_TTCAN_TTLGT)	32	R	0000_0000h	48.4.5.57/ 2013
138	TT Cycle Time and Count register (M_TTCAN_TTCTC)	32	R	003F_0000h	48.4.5.58/ 2013
13C	TT Capture Time register (M_TTCAN_TTCPT)	32	R	0000_0000h	48.4.5.59/ 2014
140	TT Cycle Sync Mark register (M_TTCAN_TTCSM)	32	R	0000_0000h	48.4.5.60/ 2014

48.4.5.1 Core Release Register (M_TTCAN_CREL)

NOTE

The coding of revisions depends on the module version used in the device.

Address: 0h base + 0h offset = 0h

Bit	0–3	4–7	8–11	12–15	16–23	24–31
R	REL[3:0]	STEP[3:0]	SUBSTEP[3:0]	YEAR[3:0]	MON[7:0]	DAY[7:0]
W						
Reset	x*	x*	x*	x*	x*	x*

M_TTCAN_CREL field descriptions

Field	Description
0–3 REL[3:0]	Core Release One digit, BCD.
4–7 STEP[3:0]	Step of Core Release One digit, BCD.
8–11 SUBSTEP[3:0]	Sub-step of Core Release One digit, BCD.
12–15 YEAR[3:0]	Time Stamp Year One digit, BCD. .
16–23 MON[7:0]	Time Stamp Month Two digits, BCD.
24–31 DAY[7:0]	Time Stamp Day Two digits, BCD.

48.4.5.2 Endian Register (M_TTCAN_ENDN)

Address: 0h base + 4h offset = 4h

Bit	0–15	16–31
R	ETV	
W		
Reset	1 0 0 0 0 1 1 1 0 1 1 0 0 1 0 1	0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1

M_TTCAN_ENDN field descriptions

Field	Description
0–31 ETV	Endianness Test Value The endianness test value is 0x87654321.

48.4.5.3 Fast Bit Timing and Prescaler Register (M_TTCAN_FBTP)

The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 M_TTCAN clock periods.

tq = (FBRP + 1) M_TTCAN clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. FTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [FTSEG1 + FTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

NOTE

With a M_TTCAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_TTCAN for a fast bit rate of 500 kbit/s.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TDCO				TDC	0	FBRP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				FTSEG1				0	FTSEG2			0		FSJW	
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

M_TTCAN_FBTP field descriptions

Field	Description
0–3 TDCO	Transceiver Delay Compensation Offset

Table continues on the next page...

M_TTCAN_FBTP field descriptions (continued)

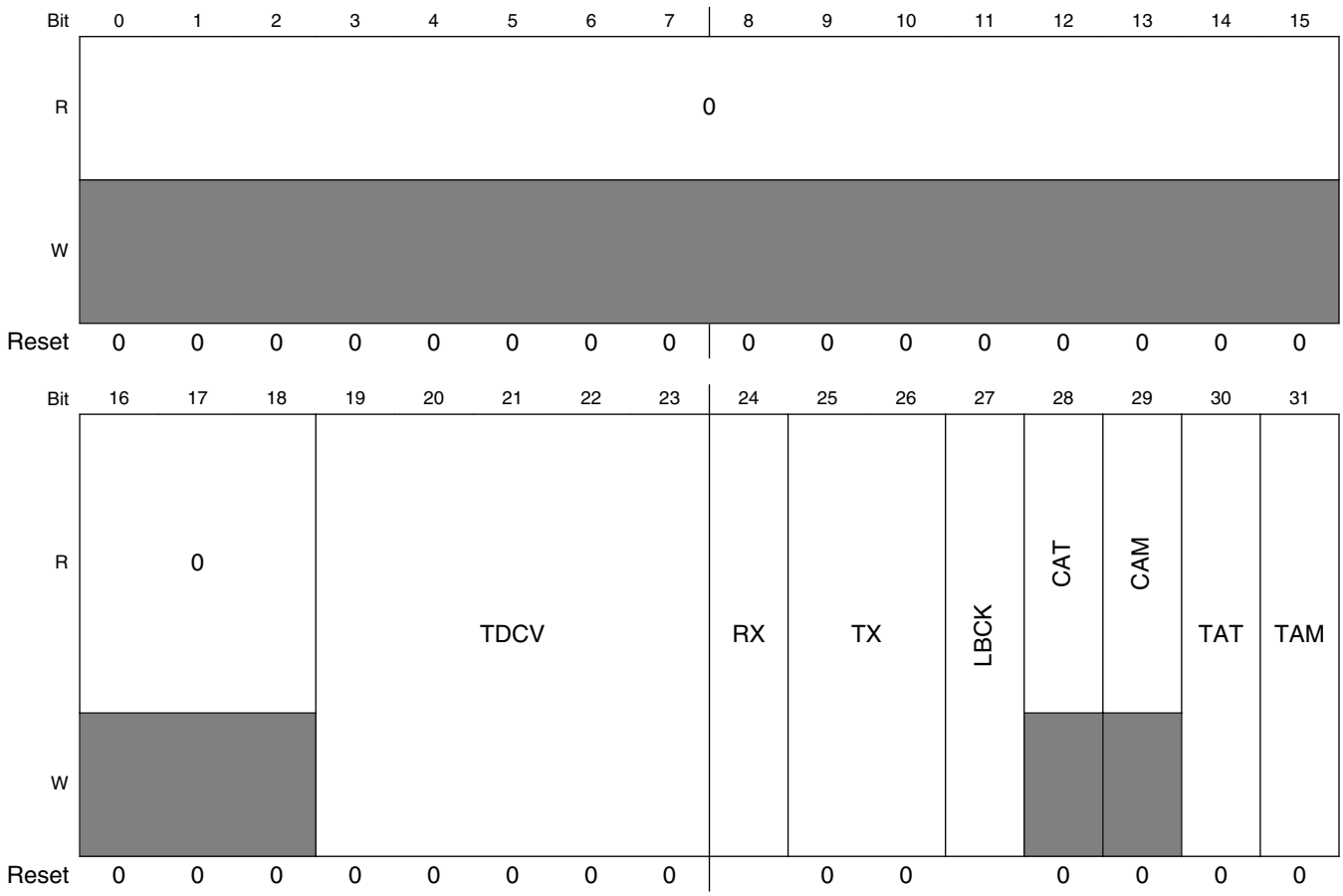
Field	Description
	(0x0-0xF) - Offset value defining the distance between the measured delay from M_TTCAN transmit to M_TTCAN receive and the secondary sample point. Valid values are 0 to 15 M_TTCAN clock periods.
4 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FBRP	Fast Baud Rate Prescaler (0x000-0x3FF)- The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 FTSEG1	Fast time segment before sample point (0x1-0xF) Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 FTSEG2	Fast time segment before sample point (0x0-0x7) Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FSJW	Fast (Re) Synchronization Jump Width 0x0-0x3 Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

48.4.5.4 Test Register (M_TTCAN_TEST)

Write access to the Test Register has to be enabled by setting bit CCCR[TEST] to '1'. All Test Register functions are set to their reset values when bit CCCR[TEST] is reset.

Loop Back mode and software control of Tx pin are hardware test modes. Programming of TX to values other than "00" may disturb the message transfer on the CAN bus.

Address: 0h base + 10h offset = 10h



M_TTCAN_TEST field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 TDCV	Transceiver Delay Compensation Value (0x00-0x1F)- Position of the secondary sample point, defined by the sum of the measured delay from M_TTCAN transmit and to receive and FBTP.TDCO. Valid values are 0 to 31 M_TTCAN core clock periods.
24 RX	Receive Pin Monitors the actual value of transmit pin 0 The CAN bus is dominant 1 The CAN bus is recessive
25–26 TX	Control of Transmit Pin **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 Reset value TX is controlled by the CAN Core, updated at the end of the CAN bit time 01 Sample Point can be monitored at pin

Table continues on the next page...

M_TTCAN_TEST field descriptions (continued)

Field	Description
	10 Dominant ('0') level at pin 11 Recessive ('1') at pin
27 LBCK	Loop Back mode **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (see Test modes)
28 CAT	Check ASC Transmit Control Monitors level at M_TTCAN ASC transmit control. 0 Output M_TTCAN ASC transmit control pin = ‘0’ 1 Output M_TTCAN ASC transmit control pin = ‘1’
29 CAM	Check ASC Multiplexer Control Monitors level at M_TTCAN ASC multiplexer control. 0 Output M_TTCAN ASC multiplexer control pin = ‘0’ 1 Output M_TTCAN ASC multiplexer control pin = ‘1’
30 TAT	Test ASC Transmit Control Controls output ASC Transmit Control pin in test mode, ORed with the signal from the FSE. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Level at M_TTCAN ASC transmit control pin controlled by FSE 1 Level at M_TTCAN ASC transmit control pin = ‘1’
31 TAM	Test ASC Multiplexer Control Controls output M_TTCAN ASC multiplexer control in test mode, ORed with the signal from the FSE. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Level at M_TTCAN ASC multiplexer control pin controlled by FSE 1 Level at M_TTCAN ASC multiplexer control pin = ‘1’

48.4.5.5 RAM Watchdog Register (M_TTCAN_RWD)

The RAM Watchdog monitors the READY output of the Message RAM . A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the RWD[WDC] bits. The counter is reloaded with RWD[WDC] bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag IR[WDI] bit is set. The RAM Watchdog Counter is clocked by the Host clock .

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																WDV								WDC							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_RWD field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 WDV	Watchdog Value. Actual Message RAM Watchdog Counter Value.
24–31 WDC	Watchdog Configuration. Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.6 CC Control Register (M_TTCAN_CCCR)

NOTE

Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		FDBS	FDO	CMR		CME		TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

M_TTCAN_CCCR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 FDBS	Long Frame Mode Active CAN FD Bit Rate Switching 0 This node transmits no frames with bit rate switching 1 This node transmits all frames (excl. remote frames) with bit rate switching
19 FDO	Fast Frame Mode Active CAN FD Operation 0 This node transmits all frames in CAN format according to ISO11898-1 1 This node transmits all frames (excl. remote frames) in CAN FD format
20–21 CMR	CAN Mode Request A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to ‘00’ and the status flags FDBS and FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1. 00 unchanged 01 Request CAN FD operation

Table continues on the next page...

M_TTCAN_CCCR field descriptions (continued)

Field	Description
	10 Request CAN FD operation with bit rate switching 11 Request CAN operation according ISO11898-1
22–23 CME	CAN Mode Enable 00 CAN operation according to ISO11898-1 enabled 01 Long Frame Mode enabled 10 Long + Fast Frame Mode enabled 11 CAN FD operation with bit rate switching enabled **NOTE:** When CME = ‘00’, received frames are strictly interpreted according to ISO11898-1, which leads to the transmission of an error frame when receiving a CAN FD frame. In case CME = ‘01’, transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = ‘10’/‘11’, transmission and reception of long and fast CAN FD frames is enabled.
24 TEST	Test Enable Mode **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle. .
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_TTCAN may be set in power down by stopping host clock and core clock
29 ASM	ASM Restricted Operation Mode The Restricted Operation Mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame. In the optional Restricted Operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication.

Table continues on the next page...

M_TTCAN_CCCR field descriptions (continued)

Field	Description
	The error counters are not incremented. Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR[INIT] = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started

48.4.5.7 Bit Timing and Prescaler Register (M_TTCAN_BTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programed in the range of [4....81] time This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programed in the range of 4 to 81 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 M_TTCAN core clock periods.

tq = (BRP + 1) M_TTCAN core clock period.

TSEG1 is the sum of Prop_Seg and Phase_Seg1. TSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

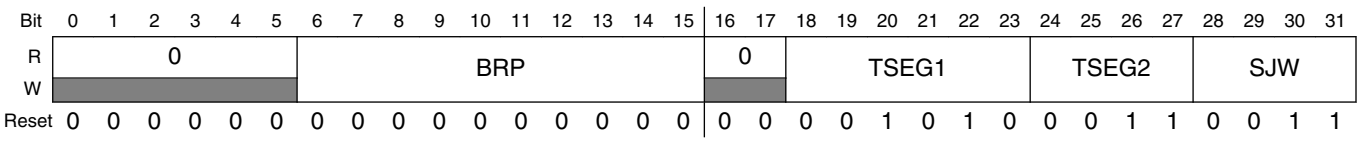
NOTE

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

NOTE

With a M_TTCAN clock of 8 MHz,the reset value of 0x000_00A33 configures the M_TTCAN for a bit rate of 500 kbit/s.

Address: 0h base + 1Ch offset = 1Ch

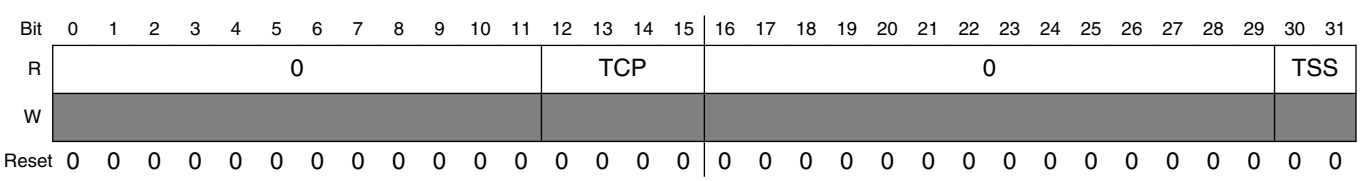


M_TTCAN_BTP field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 BRP	Baud Rate Prescaler (0x000-0x3FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TSEG1	The time segment before the sample point (0x0–0x3F) Valid values are 0 to 63. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
24–27 TSEG2	Time segment after sample point (0x0-0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
28–31 SJW	(Re) Synchronization Jump Width (0x0-0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value prog rammed here is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.8 Timestamp Counter Configuration Register (M_TTCAN_TSCC)

Address: 0h base + 20h offset = 20h



M_TTCAN_TSCC field descriptions

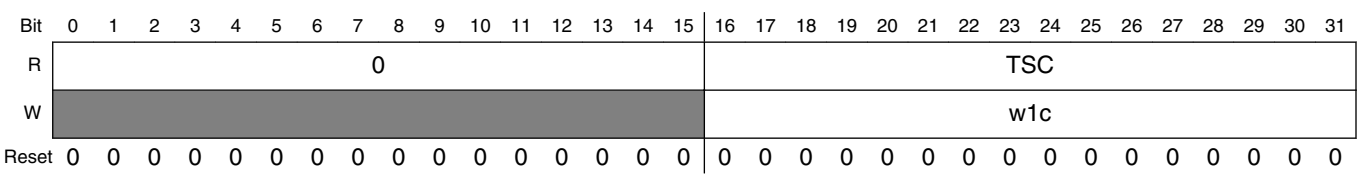
Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 TCP	Timestamp Counter Prescaler 0x0-0xF Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". **NOTE:** In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase.
16–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 TSS	Timestamp Select **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to TCP 10 Reserved 11 Same as "00"

48.4.5.9 Timestamp Counter Value Register (M_TTCAN_TSCV)

NOTE

A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

Address: 0h base + 24h offset = 24h



M_TTCAN_TSCV field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TSC	Timestamp Counter.The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = ‘01’, the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC[TCP]. A wrap around sets interrupt flag IR[TSW]. Write access resets the counter to zero. A write access has no impact.

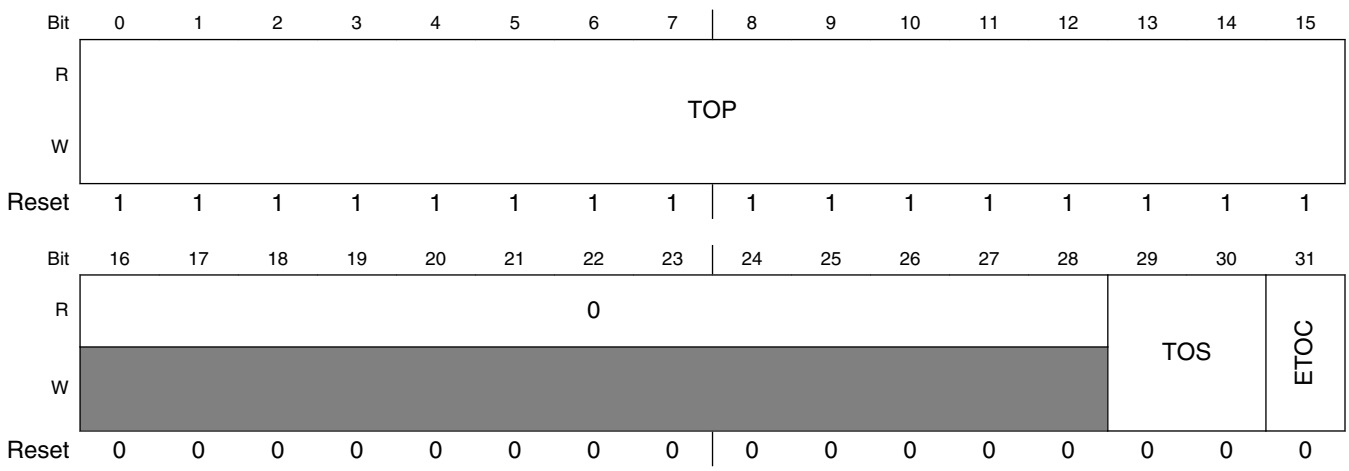
M_TTCAN_TSCV field descriptions (continued)

Field	Description

48.4.5.10 Timeout Counter Configuration Register (M_TTCAN_TOCC)

For a description of the Timeout Counter see Test modes .

Address: 0h base + 28h offset = 28h



M_TTCAN_TOCC field descriptions

Field	Description
0–15 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 TOS	Timeout Select. When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter

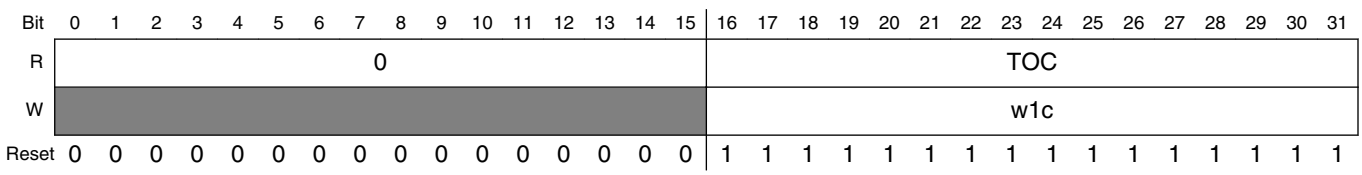
Table continues on the next page...

M_TTCAN_TOCC field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". **NOTE:** For use of timeout function with CAN FD, see section "Timeout counter". 0 Timeout Counter disabled 1 Timeout Counter enabled

48.4.5.11 Timeout Counter Value Register (M_TTCAN_TOCV)

Address: 0h base + 2Ch offset = 2Ch

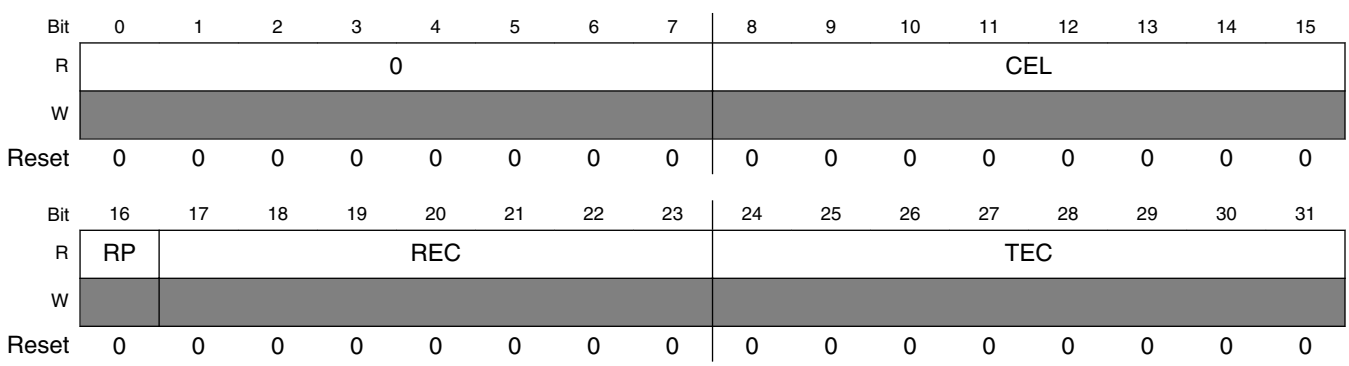


M_TTCAN_TOCV field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TOC	Timeout Counter. The Timeout Counter is decremented in multiples of CAN bit times [1:16] depending on the configuration of TSCC[TCP]. When decremented to zero, interrupt flag IR[TOO] is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC[TOS].

48.4.5.12 Error Counter Register (M_TTCAN_ECR)

Address: 0h base + 40h offset = 40h



M_TTCAN_ECR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 CEL	CAN Error Logging. The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR[ELO]. **NOTE:** This bit resets on read.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the *errorpassive* level of 128 1 The Receive Error Counter has reached the *errorpassive* level of 128
17–23 REC	Receive Error Counter. Actual state of the Receive Error Counter, values between 0 and 127
24–31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255 **NOTE:** When CCCR[ASM] is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented. This enables monitoring of collisions between CAN frames and ASC frames.

48.4.5.13 Protocol Status Register (M_TTCAN_PSR)

NOTE

Access type is RS: This register is set on read.

NOTE

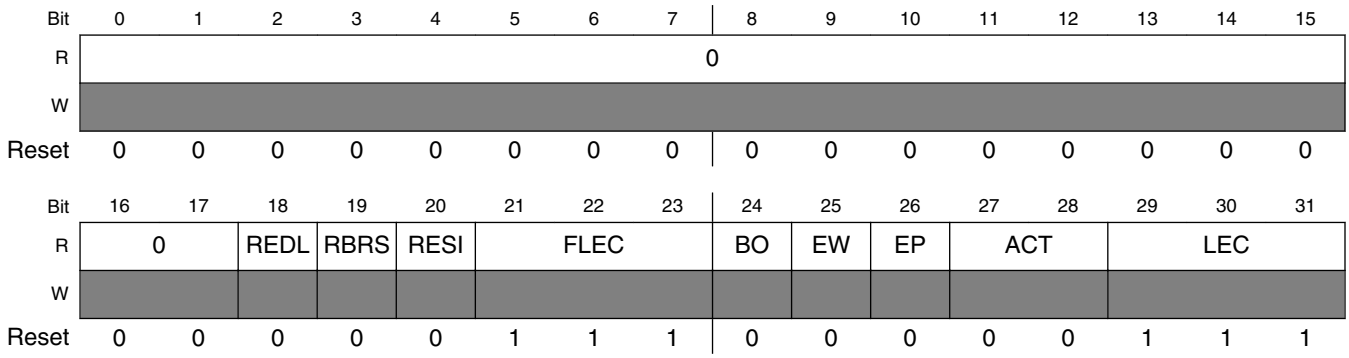
When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

NOTE

The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 x 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive

bits has been monitored, a Bit0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

Address: 0h base + 44h offset = 44h



M_TTCAN_PSR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 REDL	Received a CAN FD Message This bit is set independent of acceptance filtering. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with EDL flag set has been received **NOTE:** This bit sets on reset.
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not ha ve its BRS flag set. 1 Last received CAN FD message had its BRS flag set. **NOTE:** This bit sets on reset.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not ha ve its ESI flag set. 1 Last received CAN FD message had its ESI flag set **NOTE:** This bit sets on reset.
21–23 FLEC	Fast Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. **NOTE:** This bit resets on read.

Table continues on the next page...

M_TTCAN_PSR field descriptions (continued)

Field	Description
24 BO	Bus_Off Status 0 The M_TTCAN is not Bus_Off 1 The M_TTCAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_TTCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_TTCAN is in the Error_Passive state
27–28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing - node is synchronizing on CAN communication 01 Idle - node is neither receiver nor transmitter 10 Receiver - node is operating as receiver 11 Transmitter - node is operating as transmitter
29–31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_TTCAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a *recessive* level (bit of logical value '1'), but the monitored bus value was *dominant* . 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a *dominant* level (data or identifier bit logical value '0'), but the monitored bus value was *recessive* . During *Bus_Off* recovery this status is set each time a sequence of 11 *recessive* bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. **NOTE:** This bit resets on read.

48.4.5.14 Interrupt Register (M_TTCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	UMD	TSW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_IR field descriptions

Field	Description
0 STE	Stuff Error 0 No Stuff Error detected 1 More than 5 equal bits in a sequence occurred
1 FOE	Format Error 0 No Format Error detected 1 A fixed format part of a received frame has the wrong format
2 ACKE	Acknowledge Error 0 No Acknowledge Error detected 1 A transmitted message was not acknowledged by another node
3 BE	Bit Error 0 No Bit Error detected 1 Device wanted to send a *rec* / *dom* level, but monitored bus level was *dom* / rec
4 CRCE	CRC Error

Table continues on the next page...

M_TTCAN_IR field descriptions (continued)

Field	Description
	0 No CRC Error detected 1 Received CRC did not match the calculated CRC
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (e.g. parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (e.g. ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buff er
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 UMD	Unprocessed Message Discarded. When a new message is received while the acceptance filtering process for the previously received message has not yet completed, this message is discarded. 0 No unprocessed message discarded 1 Unprocessed message discarded

Table continues on the next page...

M_TTCAN_IR field descriptions (continued)

Field	Description
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Element Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIFO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark

Table continues on the next page...

M_TTCAN_IR field descriptions (continued)

Field	Description
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

48.4.5.15 Interrupt Enable Register (M_TTCAN_IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signaled on an interrupt line.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	UMDE	TSWE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_IE field descriptions

Field	Description
0 STEE	Stuff Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 FOEE	Format Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 ACKEE	Acknowledge Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
3 BEE	Bit Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 CRCEE	CRC Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table continues on the next page...

M_TTCAN_IE field descriptions (continued)

Field	Description
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 UMDE	Unprocessed Message Discarded Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table continues on the next page...

M_TTCAN_IE field descriptions (continued)

Field	Description
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

48.4.5.16 Interrupt Line Select Register (M_TTCAN_ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	UMDL	TSWL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_ILS field descriptions

Field	Description
0 STEL	Stuff Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
1 FOEL	Format Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
2 ACKEL	Acknowledge Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
3 BEL	Bit Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
4 CRCEL	CRC Error Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

Table continues on the next page...

M_TTCAN_ILS field descriptions (continued)

Field	Description
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
14 UMDL	UMDL: Unprocessed Message Discarded Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

Table continues on the next page...

M_TTCAN_ILS field descriptions (continued)

Field	Description
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

Table continues on the next page...

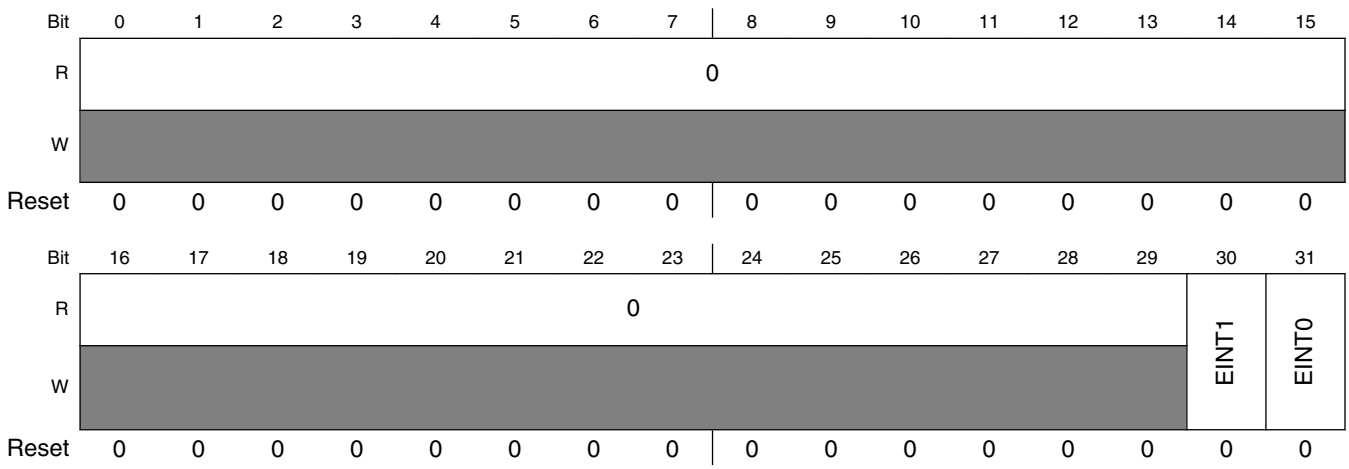
M_TTCAN_ILS field descriptions (continued)

Field	Description
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line MCAN INT0 1 Interrupt assigned to interrupt line MCAN INT1

48.4.5.17 Interrupt Line Enable Register (M_TTCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

Address: 0h base + 5Ch offset = 5Ch



M_TTCAN_ILE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled

Table continues on the next page...

M_TTCAN_ILE field descriptions (continued)

Field	Description
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

48.4.5.18 Global Filter Configuration Register (M_TTCAN_GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in Standard message ID filtering and Extended message ID filtering.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										ANFS		ANFE		RRFS	RRFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_GFC field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28–29 ANFE	Accept Non-matching Frames Extended. Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

Table continues on the next page...

M_TTCAN_GFC field descriptions (continued)

Field	Description
	00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

48.4.5.19 Standard ID Filter Configuration Register (M_TTCAN_SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LSS								FLSSA														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_SIDFC field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 LSS	List Size Standard 0 No standard Message ID filter 1-128 Number of standard Message ID filter elements >128 Values greater than 128 are interpreted as 128 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

Table continues on the next page...

M_TTCAN_SIDFC field descriptions (continued)

Field	Description
16–29 FLSSA	Filter List Standard Start Address. Start address of standard Message ID filter list (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.20 Extended ID Filter Configuration Register (M_TTCAN_XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in Figure 48-22.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									LSE							FLESA														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_XIDFC field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LSE	List Size Extended 0 No extended Message ID filter 1-64 Number of extended Message ID filter elements >64 Values greater than 64 are interpreted as 64 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–29 FLESA	Filter List Extended Start Address. Start address of extended Message ID filter list (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.21 Extended ID and Mask Register (M_TTCAN_XIDAM)

Address: 0h base + 90h offset = 90h

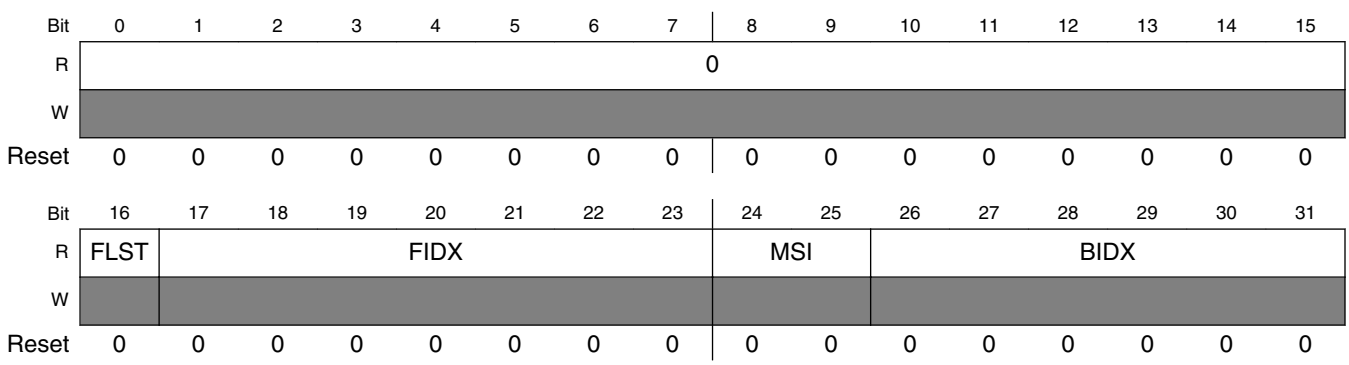
M_TTCAN_XIDAM field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–31 EIDM	Extended ID Mask. For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.22 High Priority Message Status Register (M_TTCAN_HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address: 0h base + 94h offset = 94h



M_TTCAN_HPMS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_TTCAN_HPMS field descriptions (continued)

Field	Description
16 FLST	Filter List. Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17–23 FIDX	Filter Index. Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24–25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26–31 BIDX	Buffer Index. Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.

48.4.5.23 New Data 1 Register (M_TTCAN_NDAT1)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ND1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_NDAT1 field descriptions

Field	Description
0–31 ND1	New Data[0:31] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register.

48.4.5.24 New Data 2 Register (M_TTCAN_NDAT2)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ND2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_NDAT2 field descriptions

Field	Description
0–31 ND2	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

48.4.5.25 Rx FIFO 0 Configuration Register (M_TTCAN_RXF0C)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F0WM							0	F0S						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FOSA														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_RXF0C field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 F0WM	Rx FIFO 0 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) >64 Watermark interrupt disabled **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 F0S	Rx FIFO 0 Size 0 No Rx FIFO 0 1-64 Number of Rx FIFO 0 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

Table continues on the next page...

M_TTCAN_RXF0C field descriptions (continued)

Field	Description
16–29 FOSA	Rx FIFO 0 Start Address. Start address of Rx FIFO 0 in Message RAM (32-bit word address, Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.26 Rx FIFO 0 Status Register (M_TTCAN_RXF0S)

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						RF0L	F0F	0		F0PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_RXF0S field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RF0L	Rx FIFO 0 Message Lost. This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 F0PI	Rx FIFO 0 Put Index. Rx FIFO 0 write index pointer, range 0 to 63.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 F0GI	Rx FIFO 0 Get Index. Rx FIFO 0 read index pointer, range 0 to 63.

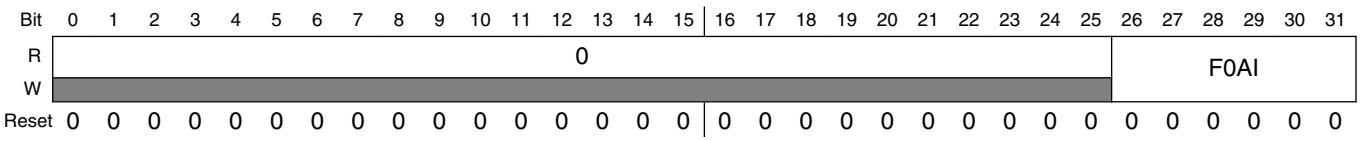
Table continues on the next page...

M_TTCAN_RXF0S field descriptions (continued)

Field	Description
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 F0FL	Rx FIFO 0 Fill Level. Number of elements stored in Rx FIFO 0, range 0 to 64.

48.4.5.27 Rx FIFO 0 Acknowledge Register (M_TTCAN_RXF0A)

Address: 0h base + A8h offset = A8h

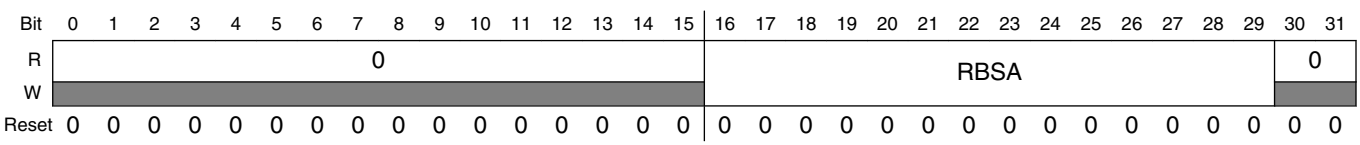


M_TTCAN_RXF0A field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F0AI	Rx FIFO 0 Acknowledge Index. After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

48.4.5.28 Rx Buffer Configuration Register (M_TTCAN_RXBC)

Address: 0h base + ACh offset = ACh

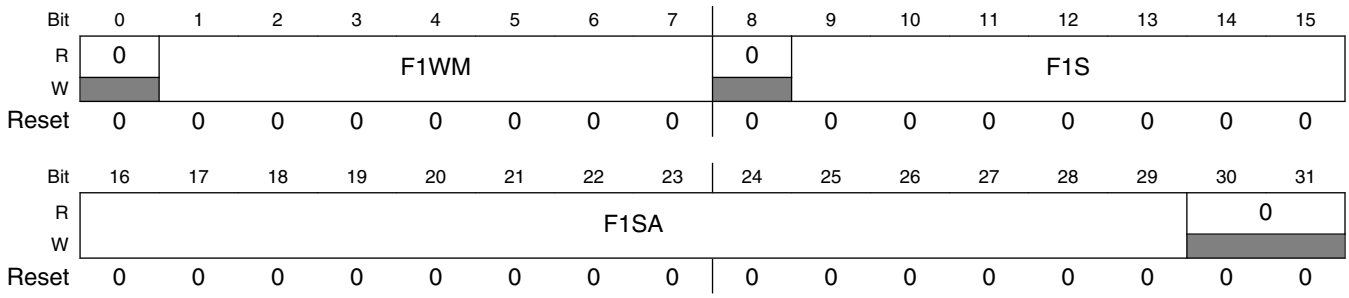


M_TTCAN_RXBC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.29 Rx FIFO 1 Configuration Register (M_TTCAN_RXF1C)

Address: 0h base + B0h offset = B0h



M_TTCAN_RXF1C field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 F1WM	Rx FIFO 1 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 1 watermark interrupt (IR[RF1W]) >64 Watermark interrupt disabled **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 F1S	Rx FIFO 1 Size 0 No Rx FIFO 1 1-64 Number of Rx FIFO 1 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–29 F1SA	Rx FIFO 1 Start Address. Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.30 Rx FIFO 1 Status Register (M_TTCAN_RXF1S)

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0				RF1L	F1F	0		F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

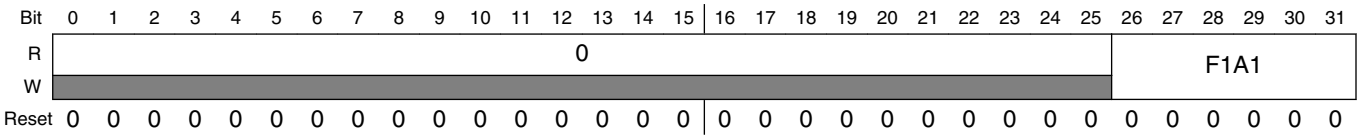
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_RXF1S field descriptions

Field	Description
0–1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
2–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RF1L	Rx FIFO 1 Message Lost. This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 F1PI	Rx FIFO 1 Put Index. Rx FIFO 1 write index pointer, range 0 to 63.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 F1GI	Rx FIFO 1 Get Index. Rx FIFO 1 read index pointer, range 0 to 63.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 F1FL	Rx FIFO 1 Fill Level. Number of elements stored in Rx FIFO 1, range 0 to 64.

48.4.5.31 Rx FIFO 1 Acknowledge register (M_TTCAN_RXF1A)

Address: 0h base + B8h offset = B8h



M_TTCAN_RXF1A field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F1A1	Rx FIFO 1 Acknowledge Index. After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

48.4.5.32 Tx Buffer Configuration register (M_TTCAN_TXBC)

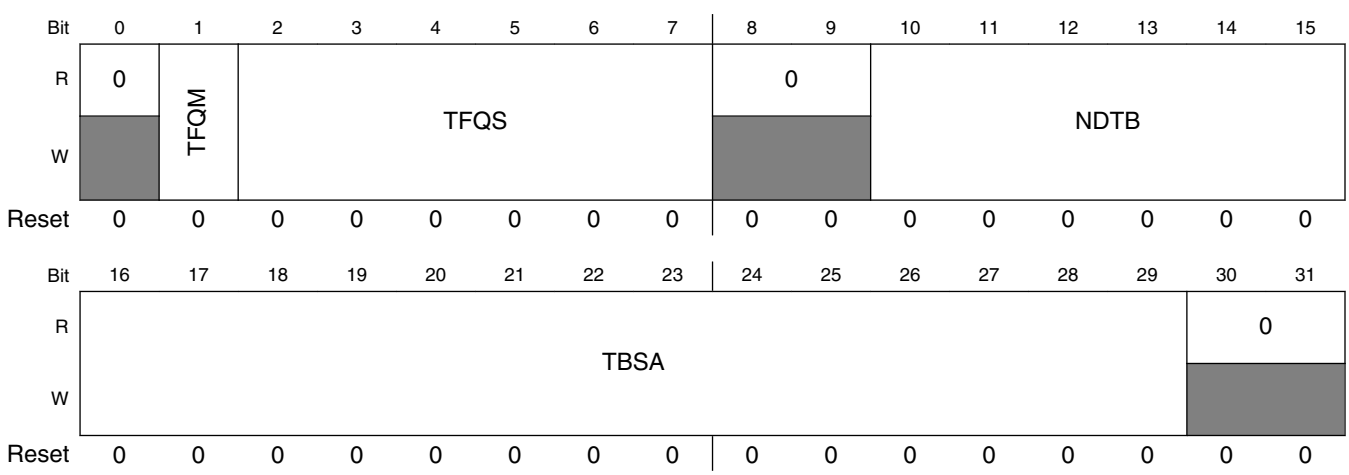
NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

NOTE

Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

Address: 0h base + C0h offset = C0h



M_TTCAN_TXBC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TFQM	Tx FIFO/Queue Mode. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Tx FIFO operation 1 Tx Queue operation
2–7 TFQS	Transmit FIFO/Queue Size. 0 No Tx FIFO/Queue 1-32 Number of Tx Buffers used for Tx FIFO/Queue >32 Values greater than 32 are interpreted as 32 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 NDTB	Number of Dedicated Transmit Buffers. 0 No Dedicated Tx Buffers 1-32 Number of Dedicated Tx Buffers >32 Values greater than 32 are interpreted as 32 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–29 TBSA	Tx Buffers Start Address. Start address of Tx Buffers section in Message RAM (32-bit word address, Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.33 Tx FIFO/Queue Status register (M_TTCAN_TXFQS)

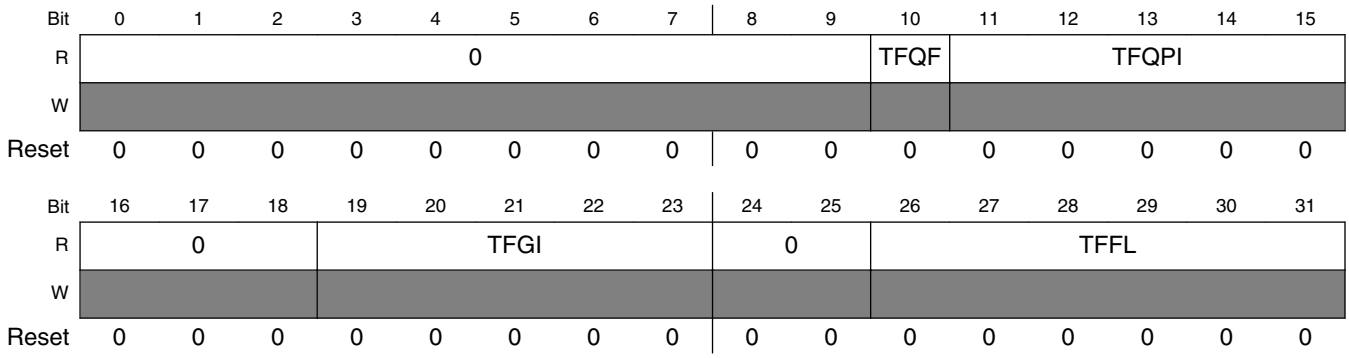
The Tx FIFO/Queue status is related to the pending Tx requests listed in register TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (TXBRP not yet updated).

NOTE

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put Index indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.

For example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Address: 0h base + C4h offset = C4h



M_TTCAN_TXFQS field descriptions

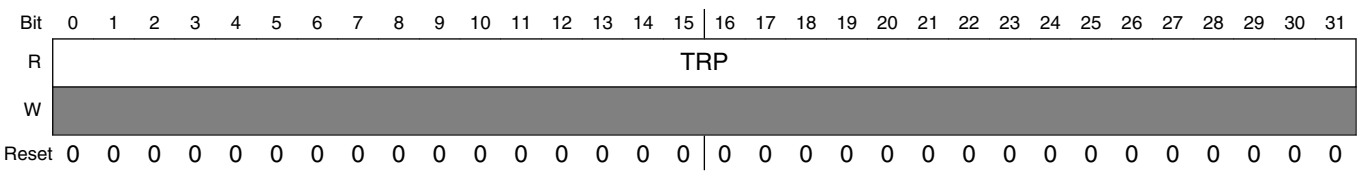
Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 TFQF	Tx FIFO/Queue Full. 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11–15 TFQPI	Tx FIFO/Queue Put Index. Tx FIFO/Queue write index pointer, range 0 to 31.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 TFGI	Tx FIFO Get Index. Tx FIFO read index pointer, range 0 to 31. For internal use only. Read as zero when Tx Queue operation is configured (TXBC.TFQM = â€˜1â€™).
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFFL	Tx FIFO Free Level. Number of consecutive free Tx FIFO elements , range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = '1')

48.4.5.34 Tx Buffer Request Pending register (M_TTCAN_TXBRP)

NOTE

TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

Address: 0h base + CCh offset = CCh



M_TTCAN_TXBRP field descriptions

Field	Description
0–31 TRP	Transmission Request Pending. Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (see Debug on CAN Support) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF • After successful transmission together with the corresponding TXBTO bit • When the transmission has not yet been started at the point of cancellation • When the transmission has been aborted due to lost arbitration • When an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

48.4.5.35 Tx Buffer Add Request register (M_TTCAN_TXBAR)

NOTE

If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXBAR field descriptions

Field	Description
0–31 AR	Add Request. Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

48.4.5.36 Tx Buffer Cancellation Request register (M_TTCAN_TXBCR)

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXBCR field descriptions

Field	Description
0–31 CR	Cancellation Request. Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

48.4.5.37 Tx Buffer Transmission Occurred register (M_TTCAN_TXBTO)

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXBTO field descriptions

Field	Description
0–31 TO	Transmission Occurred. Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

48.4.5.38 Tx Buffer Cancellation Finished register (M_TTCAN_TXBCF)

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CF																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXBCF field descriptions

Field	Description
0–31 CF	Cancellation Finished. Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

48.4.5.39 Tx Buffer Transmission Interrupt Enable register (M_TTCAN_TXBTIE)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXBTIE field descriptions

Field	Description
0–31 TIE	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

48.4.5.40 Tx Buffer Cancellation Finished Interrupt Enable register (M_TTCAN_TXBCIE)

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CFIE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXBCIE field descriptions

Field	Description
0–31 CFIE	Cancellation Finished Interrupt Enable. Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

48.4.5.41 Tx Event FIFO Configuration register (M_TTCAN_TXEFC)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

Address: 0h base + F0h offset = F0h

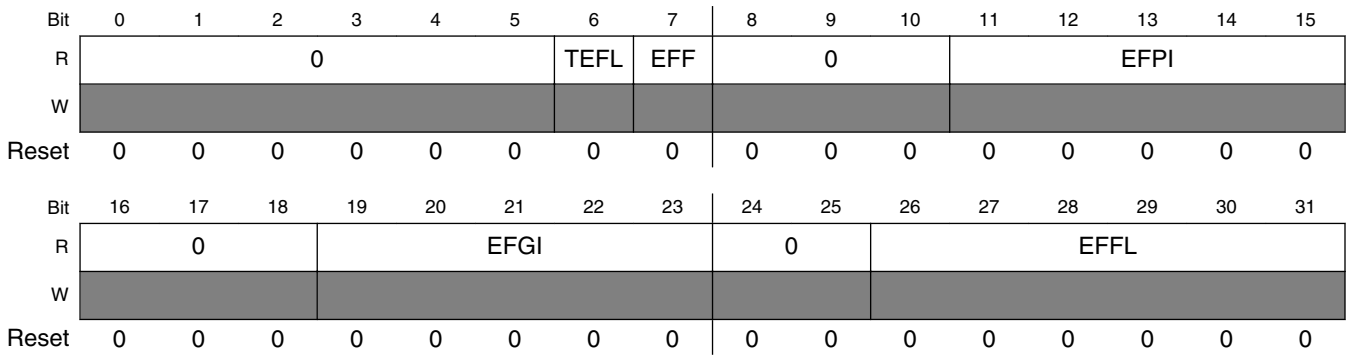
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		EFWM						0		EFS						EFSA														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TXEFC field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 EFWM	Event FIFO Watermark 0 Watermark interrupt disabled 1-32 Level for Tx Event FIFO watermark interrupt (IR[TEFW]) >32 Watermark interrupt disabled **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 EFS	Event FIFO Size. 0 Tx Event FIFO disabled 1-32 Number of Tx Event FIFO elements >32 Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS-1 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–29 EFSA	Event FIFO Start Address Start address of Tx Event FIFO in Message RAM (32-bit word address, Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.42 Tx Event FIFO Status register (M_TTCAN_TXEFS)

Address: 0h base + F4h offset = F4h



M_TTCAN_TXEFS field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 TEFL	Tx Event FIFO Element Lost. This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full. 0 Tx Event FIFO not full 1 Tx Event FIFO full
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 EFPI	Event FIFO Put Index. Tx Event FIFO write index pointer, range 0 to 31.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 EFGI	Event FIFO Get Index. Tx Event FIFO read index pointer, range 0 to 31.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 EFFL	Event FIFO Fill Level. Number of elements stored in Tx Event FIFO, range 0 to 32.

48.4.5.43 Tx Event FIFO Acknowledge register (M_TTCAN_TXEFA)

Address: 0h base + F8h offset = F8h

Bit	0–26	27–31
R	0	EFAI
W		
Reset	0	0

M_TTCAN_TXEFA field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 EFAI	Event FIFO Acknowledge Index. After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

48.4.5.44 TT Trigger Memory Configuration register (M_TTCAN_TTTMC)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

Address: 0h base + 100h offset = 100h

Bit	0–8	9–15	16–29	30–31
R	0	TME	TMSA	0
W				
Reset	0	0	0	0

M_TTCAN_TTTMC field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 TME	Trigger Memory Elements. 0 No Trigger Memory 1-64 Number of Trigger Memory elements >64 Values greater than 64 are interpreted as 64

Table continues on the next page...

M_TTCAN_TTTMC field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–29 TMSA	Trigger Memory Start Address. Start address of Trigger Memory in Message RAM (32-bit word address, Message RAM). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5.45 TT Reference Message Configuration register (M_TTCAN_TTRMC)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RMPS	XTD	0	RID												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTRMC field descriptions

Field	Description
0 RMPS	Reference Message Payload Select. Ignored in case of time slaves. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

Table continues on the next page...

M_TTCAN_TTRMC field descriptions (continued)

Field	Description
	0 Reference message has no additional payload 1 The following elements are taken from Tx Buffer 0: Message Marker MM, Event FIFO Control EFC, Data Length Code DLC, Data Bytes DB (Level 1: bytes 2-8, Level 0, 2: bytes 5-8)
1 XTD	Extended Identifier **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 11-bit standard identifier 1 11-bit standard identifier
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–31 RID	Reference Identifier. Identifier transmitted with Reference message and used for Reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.46 TT Operation Configuration register (M_TTCAN_TTOCF)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					EVTP	ECC	EGTF	AWL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EECS	IRTO							LDSDL			TM	GEN	0	OM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTOCF field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 EVTP	Event Trigger Polarity.

Table continues on the next page...

M_TTCAN_TTOCF field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Rising edge trigger 1 Falling edge trigger
6 ECC	Enable Clock Calibration. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Automatic clock calibration in M_TTCAN Level 0, 2 is disabled 1 Automatic clock calibration in M_TTCAN Level 0, 2 is enabled
7 EGTF	Enable Global Time Filtering. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Global time filtering in M_TTCANM_TTCAN Level 0, 2 is disabled 1 Global time filtering in M_TTCAN Level 0, 2 is enabled
8–15 AWL	Application Watchdog Limit. The application watchdog can be disabled by programming AWL to 0x00. 0x00-FF: Maximum time after which the application has to ser ve the application watchdog. The application watchdog is incremented once each 256 NTUs. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16 EECS	Enable External Clock Synchronization. If enabled, TUR configuration (TURCF[NCL] only) may be updated during M_TTCAN operation. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 External clock synchronization in M_TTCAN Level 0,2 disabled 1 External clock synchronization in M_TTCAN Level 0,2 enabled
17–23 IRTO	Initial Reference Trigger Offset. 0x00-7F Positive offset, range from 0 to 127 **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
24–26 LDSDL	LD of Synchronization Deviation Limit. The Synchronization Deviation Limit SDL is configured by its dual logarithm LDSDL with SDL = $2^{(\mathbf{LDSDL}+5)}$. It should not exceed the clock tolerance given by the CAN bit timing configuration. 0x0-7 LD of Synchronization Deviation Limit (SDL ≤ 32…4096) **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
27 TM	Time Master. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Time Master function disabled 1 Potential Time Master

Table continues on the next page...

M_TTCAN_TTOCF field descriptions (continued)

Field	Description
28 GEN	Gap Enable. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Strictly time-triggered operation 1 External event-synchronized time-triggered operation
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 OM	Operation Mode. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 Event-driven CAN communication, default 01 M_TTCAN level 1 10 M_TTCAN level 2 11 M_TTCAN level 0

48.4.5.47 TT Matrix Limits register (M_TTCAN_TTMLM)

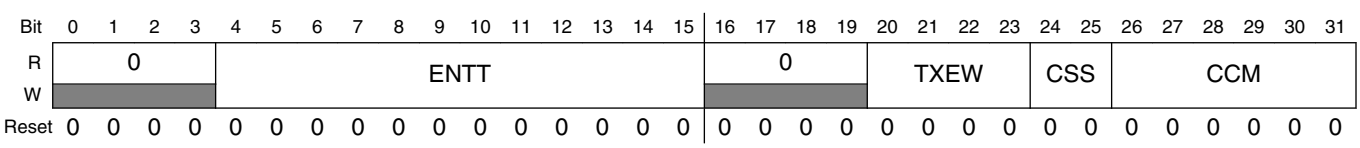
NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

NOTE

ISO 11898-4, Section 5.2.1 requires, that only the listed cycle count values are configured. Other values are possible but may lead to inconsistent matrix cycles.

Address: 0h base + 10Ch offset = 10Ch



M_TTCAN_TTMLM field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 ENTT	Expected Number of Tx Triggers 0x000-FFF Expected number of Tx Triggers in one Matrix Cycle

Table continues on the next page...

M_TTCAN_TTMLM field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 TXEW	Tx Enable Window. 0x0-F Length of Tx enable window, 1-16 NTU cycles **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
24–25 CSS	Cycle Start Synchronization Enables sync pulse output at M_TTCAN start of cycle pin. 00 No sync pulse 01 Sync pulse at start of basic cycle 10 Sync pulse at start of matrix cycle 11 Reserved **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
26–31 CCM	Cycle Count Max. 0x00 1 Basic Cycle per Matrix Cycle 0x01 2 Basic Cycles per Matrix Cycle 0x03 4 Basic Cycles per Matrix Cycle 0x07 8 Basic Cycles per Matrix Cycle 0x0F 16 Basic Cycles per Matrix Cycle 0x1F 32 Basic Cycles per Matrix Cycle 0x3F 64 Basic Cycles per Matrix Cycle others Reserved **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.48 TUR Configuration register (M_TTCAN_TURCF)

NOTE

Protected write (P) register which means that the write access to this register is possible only when the CCCR[CCE] and CCCR[INIT] register fields are set to "1".

The length of the NTU is given by: NTU = CAN Clock Period x NC/DC

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC is 0x10000:0x1FFFF. The value configured by NCL is the initial value for TURNA[NAV[15:0]]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.

Level 1: NC >= 4 x DC and NTU = CAN bit time

Level 0, 2: NC >=8 x DC

The actual value of TUR may be changed by the clock drift compensation function of M_TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. DC will not be changed by the automatic drift compensation, TURNA[NAV] may be adjusted around NC in the range of the Synchronization Deviation Limit given by TTOCF[LDSDL]. NC and DC should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

NOTE

If NC < 7 x DC in M_TTCAN Level 1, then it is required that subsequent Time Marks in the Trigger Memory must differ by at least two NTU.

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELT	0	DC													
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TURCF field descriptions

Field	Description
0 ELT	Enable Local Time. Please note that the local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. TURCF[DC] is locked when TURCF[ELT] = '1'. If ELT is written to '0', the readable value will stay at '1' until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Local time is stopped, default 1 Local time is enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 DC	Denominator Configuration.

Table continues on the next page...

M_TTCAN_TURCF field descriptions (continued)

Field	Description
	0x0000 Illegal value 0x0001-3FFF Denominator Configuration **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
16–31 NCL	Numerator Configuration Low. Write access to the TUR Numerator Configuration Low is only possible during configuration with TURCF[ELT] = '0' or if TTOCF[EECS] (external clock synchronization enabled) is set. When a new value for NCL is written outside TT Configuration Mode, the new value takes effect when TTOST.WECS is cleared to '0'. NCL is locked TTOST[WECS] is '1'. 0x0000-FFFF Numerator Configuration Low **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.49 TT Operation Control register (M_TTCAN_TTOCN)

NOTE

Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LCKC	0	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC		RTIE	SWS		SWP	ECS	SGT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTOCN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LCKC	TT Operation Control Register Locked. Set by a write access to register TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain. 0 Write access to TTOCN enabled 1 Write access to TTOCN locked

Table continues on the next page...

M_TTCAN_TTOCN field descriptions (continued)

Field	Description
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ESCN	External Synchronization Control. If enabled the M_TTCAN synchronizes its cycle time phase to an external event signaled by a rising edge at Event Trigger pin (Synchronization to external time schedule). **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 External synchronization disabled 1 External synchronization enabled
19 NIG	Next is Gap. This bit can only be set when the M_TTCAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (TTOCF[GEN] = '1') **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 No action, reset by reception of any Reference message 1 Transmit next Reference Message with Next_is_Gap = '1'
20 TMG	Time Mark Gap. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Reset by each Reference message 1 Next Reference message started when Register Time Mark interrupt TTIR[RTMI] is activated
21 FGP	Finish Gap. Set by the CPU, reset by each reference message **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 No reference message requested 1 Application requested start of reference message
22 GCS	Gap Control Select **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Gap control independent from Event trigger 1 Gap control by input Event trigger pin
23 TTIE	Trigger Time Mark Interrupt Pulse Enable External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Schedule or In_Gap. 0 Trigger Time Mark Interrupt output m_ttcan_tmp disabled 1 Trigger Time Mark Interrupt output m_ttcan_tmp enabled
24–25 TMC	Register Time Mark Compare. **NOTE:** When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = "00", then reconfigure TTTMK, and finally set TMC to the intended time reference.

Table continues on the next page...

M_TTCAN_TTOCN field descriptions (continued)

Field	Description
	NOTE: Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 No Register Time Mark Interrupt generated 01 Register Time Mark Interrupt if Time Mark = cycle time 10 Register Time Mark Interrupt if Time Mark = local time 11 Register Time Mark Interrupt if Time Mark = global time
26 RTIE	Register Time Mark Interrupt Pulse Enable. Register time mark interrupts are configured by register TTTMK. A register time mark interrupt pulse with the length of one m_ttcan_clk period is generated when time referenced by TTOCN[TMC] (cycle, local, or global) equals TTTMK[TM], independent of the synchronization state. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Register Time Mark Interrupt output disabled 1 Register Time Mark Interrupt output enabled
27–28 SWS	Stop Watch Source. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 00 Stop Watch disabled 01 Actual value of cycle time is copied to TTCPT[SWV] 10 Actual value of local time is copied to TTCPT[SWV] 11 Actual value of global time is copied to TTCPT[SWV]
29 SWP	Stop Watch Polarity. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1". 0 Rising edge trigger 1 Falling edge trigger
30 ECS	External Clock Synchronization. Writing a '1' to ECS sets TTOST[WECS] if the node is the actual Time Master. ECS is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".
31 SGT	Set Global time. Writing a '1' to SGT sets TTOST[WGDT] if the node is the actual Time Master. SGT is reset after one Host clock period. The global time preset takes effect when the node transmits the next Reference message with the Master_Ref_Mark modified by the preset value written to TTGTP. **NOTE:** Protected write (P) bit(s) which means that write access by the bit(s) is possible only when the bit 1 [CCE] CCCR[INIT] register fields are set to "1".

48.4.5.50 TT Global Time Preset register (M_TTCAN_TTGTP)

If TTOST.WGDT is set, the next Reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = '1', presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a Reference message with Disc_Bit = '1' becomes valid or if the node is not the current Time Master. TP is locked while TTOST[WGTD] = '1' after setting TTOCN[SGT] until the Reference message with Disc_Bit = '1' becomes valid or until the node is no longer the current Time Master.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CTP																TP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTGTP field descriptions

Field	Description
0–15 CTP	Cycle Time Target Phase. CTP is write-protected while TTOCN[ESCN] or TTOST[SPL] are set (see Synchronization to external time schedule). 0x0000-FFFF Defines target value of cycle time when a rising edge of event trigger is expected
16–31 TP	Time Preset. TP is write-protected while TTOST[WGTD] is set. 0x0000-7FFF Next Master Reference Mark = Master Reference Mark + TP 0x8000 reserved 0x8001-FFFF Next Master Reference Mark = Master Reference Mark - (0x10000 - TP)

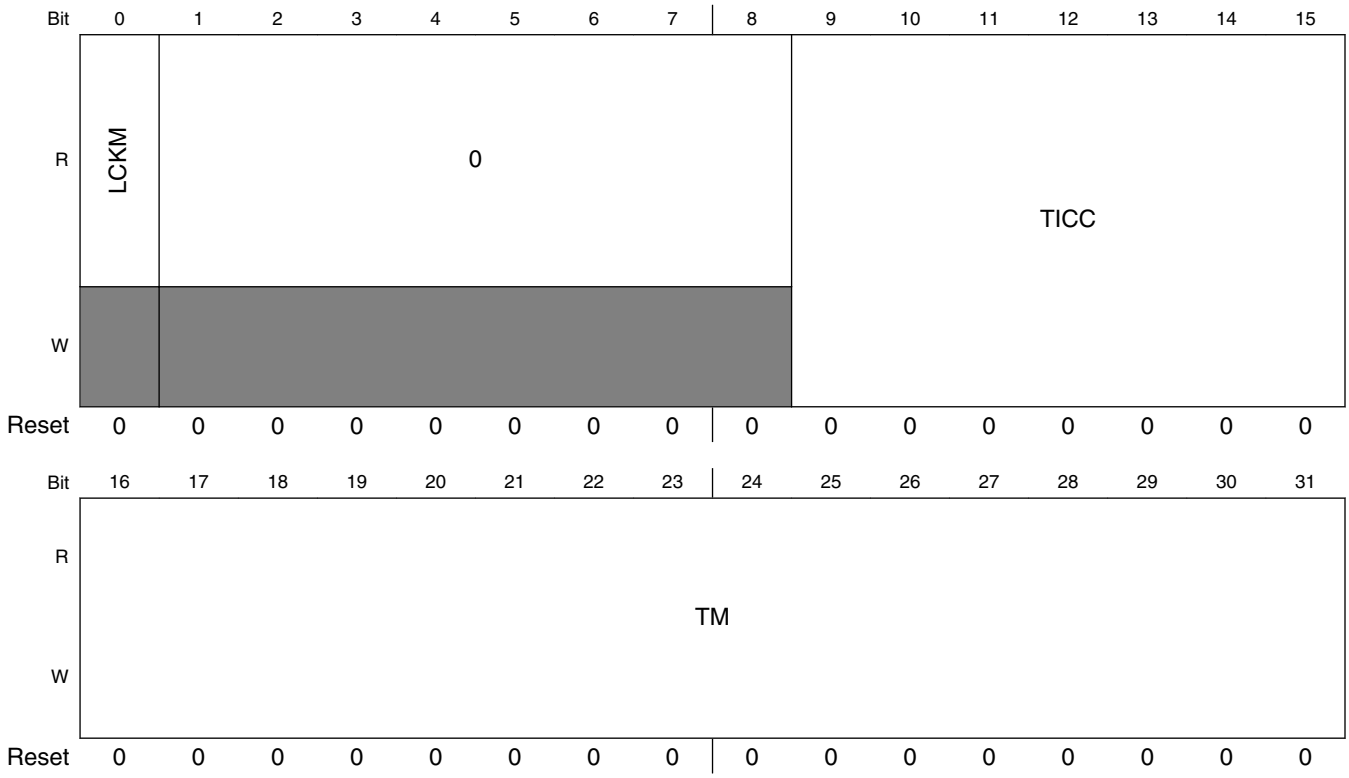
48.4.5.51 TT Time Mark register (M_TTCAN_TTTMK)

A time mark interrupt (TTIR[TMI] = '1') is generated when the time base indicated by TTOCN[TMC] (cycle time, local time, or global time) has the same value as TM.

NOTE

When using byte access to register TTTMK it is recommended to first disable the time mark compare function (TTOCN[TMC] = "00") to avoid compares on inconsistent register values.

Address: 0h base + 11Ch offset = 11Ch



M_TTCAN_TTTMK field descriptions

Field	Description
0 LCKM	TT Time Mark Register Locked. Always set by a write access to registers TTOCN. Set by write access to register TTTMK when TTOCN[TMC] "00". Reset when the registers have been synchronized into the CAN clock domain. 0 Write access to TTTMK enabled 1 Write access to TTTMK locked
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 TICC	Time Mark Cycle Code. Cycle count for which the time mark is valid. 0b000000x valid for all cycles 0b000001c valid every second cycle at cycle count mod2 = c 0b00001cc valid every fourth cycle at cycle count mod4 = cc 0b0001ccc valid every eighth cycle at cycle count mod8 = ccc 0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc 0b01ccccc valid every thirty-second cycle at cycle count mod32 = ccccc 0b1cccccc valid every sixty-fourth cycle at cycle count mod64 = cccccc
16–31 TM	Time Mark. 0x0000-FFFF Time Mark

48.4.5.52 TT Interrupt Register (M_TTCAN_TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													CER	AW	WT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWT	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTIR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CER	Configuration Error. Trigger out of order. 0 No error found in trigger list 1 Error found in trigger list
14 AW	Application Watchdog. 0 Application watchdog served in time 1 Application watchdog not served in time
15 WT	Watch Trigger. 0 No missing Reference message 1 Missing Reference message (Level 0: cycle time 0xFF00)
16 IWT	Initialization Watch Trigger. The initialization is restarted by resetting IWT. 0 No missing Reference message during system startup 1 No system startup due to missing Reference message
17 ELC	Error Level Changed. Not set when error level changed during initialization. 0 No change in error level 1 Error level changed
18 SE2	Scheduling Error 2. 0 No scheduling error 2 1 Scheduling error 2 occurred

Table continues on the next page...

M_TTCAN_TTIR field descriptions (continued)

Field	Description
19 SE1	Scheduling Error 1. 0 No scheduling error 1 1 Scheduling error 1 occurred
20 TXO	Tx Count Overflow. 0 Number of Tx Trigger as expected 1 More Tx trigger than expected in one cycle
21 TXU	Tx Count Underflow. 0 Number of Tx Trigger as expected 1 Less Tx trigger than expected in one cycle
22 GTE	Global Time Error. Synchronization deviation SD exceeds limit specified by TTOCF[LDSDL], M_TTCAN Level 0, 2 only. 0 Synchronization deviation within limit 1 Synchronization deviation exceeded limit
23 GTD	Global Time Discontinuity. 0 No discontinuity of global time 1 Discontinuity of global time
24 GTW	Global Time Wrap. 0 No global time wrap occurred 1 Global time wrap from 0xFFFF to 0x0000 occurred
25 SWE	Stop Watch Event. 0 No rising edge at stop watch trigger pin detected 1 Rising edge at stop watch trigger pin detected
26 TTMI	Trigger Time Mark Event Internal. Internal time mark events are configured by trigger memory element TMIN (see Trigger memory element). Set when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Gap or In_Schedule. 0 Time mark not reached 1 Time mark reached (Level 0: cycle time TTOCF[RTO] x 0x200)
27 RTMI	Register Time Mark Interrupt. Set when time referenced by TTOCN[TMC] (cycle, local, or global) equals TTTMK[TM], independent of the synchronization state. 0 Time mark not reached 1 Time mark reached
28 SOG	Start of Gap. 0 No reference message seen with Next_is_Gap bit set 1 Reference message with Next_is_Gap bit set becomes valid
29 CSM	Change of Synchronization Mode. 0 No change in master to slave relation or schedule synchronization 1 Master to slave relation or schedule synchronization changed, also set when TTOST[SPL] is reset
30 SMC	Start of Matrix Cycle.

Table continues on the next page...

M_TTCAN_TTIR field descriptions (continued)

Field	Description
	0 No Matrix Cycle started since bit has been reset 1 Matrix Cycle started
31 SBC	Start of Basic Cycle. 0 No Basic Cycle started since bit has been reset 1 Basic Cycle started

48.4.5.53 TT Interrupt Enable register (M_TTCAN_TTIE)

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													CERE	AWE	WTE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTIE field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CERE	Configuration Error Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
14 AWE	Application Watchdog Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
15 WTE	Watch Trigger Interrupt Enable.

Table continues on the next page...

M_TTCAN_TTIE field descriptions (continued)

Field	Description
	0 TT interrupt disabled 1 TT interrupt enabled
16 IWTE	Initialization Watch Trigger Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
17 ELCE	Change Error Level Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
18 SE2E	Scheduling Error 2 Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
19 SE1E	Scheduling Error 1 Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
20 TXOE	Tx Count Overflow Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
21 TXUE	Tx Count Underflow Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
22 GTEE	Global Time Error Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
23 GTDE	Global Time Discontinuity Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
24 GTWE	Global Time Wrap Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
25 SWEE	Stop Watch Event Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
26 TTMIE	Trigger Time Mark Event Internal Enable. 0 TT interrupt disabled 1 TT interrupt enabled
27 RTMIE	Register Time Mark Interrupt Enable.

Table continues on the next page...

M_TTCAN_TTIE field descriptions (continued)

Field	Description
	0 TT interrupt disabled 1 TT interrupt enabled
28 SOGE	Start of Gap Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
29 CSME	Change of Synchronization Mode Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
30 SMCE	Start of Matrix Cycle Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled
31 SBCE	Start of Basic Cycle Interrupt Enable. 0 TT interrupt disabled 1 TT interrupt enabled

48.4.5.54 TT Interrupt Line Select register (M_TTCAN_TTILS)

The TT Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the TT Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

0 TT interrupt assigned to interrupt line 0

1 TT interrupt assigned to interrupt line 1

Address: 0h base + 128h offset = 128h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													CERL	AWL	WTL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWE	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TTILS field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CERL	Configuration Error Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
14 AWL	Application Watchdog Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
15 WTL	Watch Trigger Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
16 IWTL	Initialization Watch Trigger Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
17 ELCL	Change Error Level Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
18 SE2L	Scheduling Error 2 Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
19 SE1L	Scheduling Error 1 Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
20 TXOL	Tx Count Overflow Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
21 TXUL	Tx Count Underflow Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
22 GTEL	Global Time Error Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
23 GTDL	Global Time Discontinuity Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
24 GTWE	Global Time Wrap Interrupt Line.

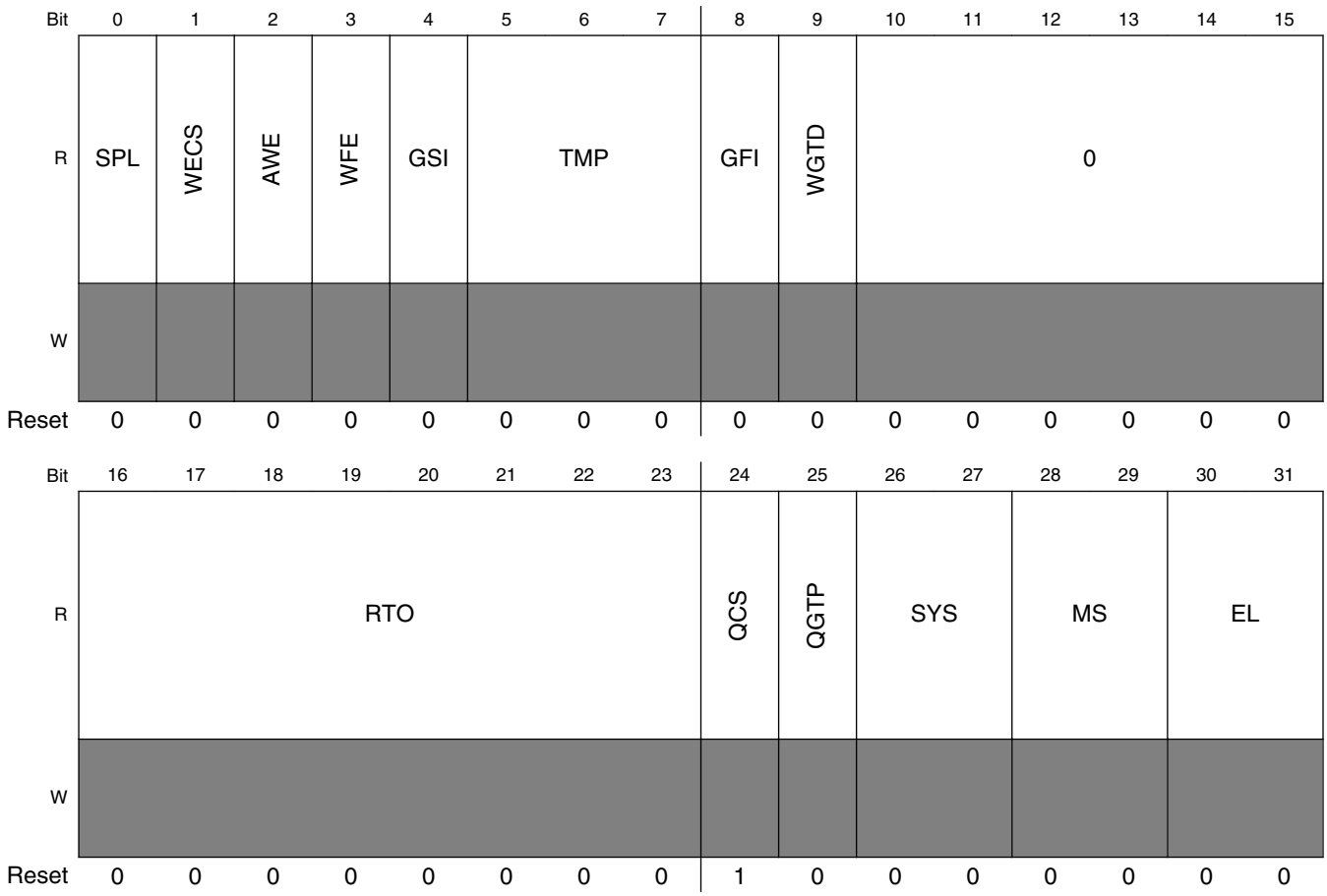
Table continues on the next page...

M_TTCAN_TTILS field descriptions (continued)

Field	Description
	0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
25 SWEL	Stop Watch Event Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
26 TTMIL	Trigger Time Mark Event Internal Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
27 RTMIL	Register Time Mark Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
28 SOGL	Start of Gap Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
29 CSML	Change of Synchronization Mode Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
30 SMCL	Start of Matrix Cycle Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
31 SBCL	Start of Basic Cycle Interrupt Line. 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1

48.4.5.55 TT Operation Status register (M_TTCAN_TTOST)

Address: 0h base + 12Ch offset = 12Ch



M_TTCAN_TTOST field descriptions

Field	Description
0 SPL	Schedule Phase Lock. The bit is valid only when external synchronization is enabled (TTOCN[ESCN] = '1'). In this case it signals that the difference between cycle time configured by TTGTP[CTP] and the cycle time at the rising edge at event trigger pin is less or equal 9 NTU (see Synchronization to external time schedule). 0 Phase outside range 1 Phase inside range
1 WECS	Wait for External Clock Synchronization. 0 No external clock synchronization pending 1 Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.
2 AWE	Application Watchdog Event. The application watchdog is served by reading TTOST. When the watchdog is not served in time, bit AWE is set, all M_TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.

Table continues on the next page...

M_TTCAN_TTOST field descriptions (continued)

Field	Description
	0 Application Watchdog served in time 1 Failed to serve Application Watchdog in time
3 WFE	Wait for Event. 0 No Gap announced, reset by a Reference Message with Next_is_Gap = '0' 1 Reference Message with Next_is_Gap = '1' received
4 GSI	Gap Started Indicator. 0 No Gap in schedule, reset by each reference message and for all time slaves 1 Gap time after Basic Cycle has started
5–7 TMP	Time Master Priority. 0x0-7 Priority of actual Time Master
8 GFI	Gap Finished Indicator. Set when the CPU writes TTOCN[FGP], or by a Time Mark Interrupt if TMG = '1', or via input pin (event trigger) if TTOCN[GCS] = '1'. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message. 0 Reset at the end of each reference message 1 Gap finished by M_TTCAN
9 WGTD	Wait for Global Time Discontinuity. 0 No global time preset pending 1 Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a Reference message with Disc_Bit = '1' or after it received a Reference message.
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 RTO	Reference Trigger Offset. The Reference Trigger Offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of-127 is reached. In case the M_TTCAN becomes Time Master (MS[1:0] = "11"), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by TTOCF[IRTO] is read. 0x00-FF Actual Reference Trigger offset value
24 QCS	Quality of Clock Speed. Only relevant in M_TTCAN Level 0 and Level 2, otherwise fixed to '1'. 0 Local clock speed not synchronized to Time Master clock speed 1 Synchronization Deviation SDL <= SDL
25 QGTP	Quality of Global Time Phase. Only relevant in M_TTCAN Level 0 and Level 2, otherwise fixed to '0'. 0 Global time not valid 1 Global time in phase with Time Master
26–27 SYS	Synchronization State. 00 Out of Synchronization 01 Synchronizing to M_TTCAN communication 10 Schedule suspended by Gap (In_Gap) 11 Synchronized to schedule (In_Schedule)
28–29 MS	Master State. 00 Master_Off, no master properties relevant

Table continues on the next page...

M_TTCAN_TTOST field descriptions (continued)

Field	Description
	01 Operating as Time Slave 10 Operating as Backup Time Master 11 Operating as current Time Master
30–31 EL	Error Level. 00 Severity 0-No Error 01 Severity 1-Warning 10 Severity 2-Error 11 Severity 3-Severe Error

48.4.5.56 TUR Numerator Actual register (M_TTCAN_TURNA)

There is no drift compensation in M_TTCAN Level 1 (NAV = NC). In M_TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated. The drift is compensated when the Synchronization Deviation (difference between NC and the calculated NAV) is not more than $2^{(\text{TTOCF[LDSDL]} + 5)}$. With TTOCF[LDSDL] <=7, this results in a maximum range for NAV of

(NC - 0x1000) <= NAV <= (NC + 0x1000).

Address: 0h base + 130h offset = 130h

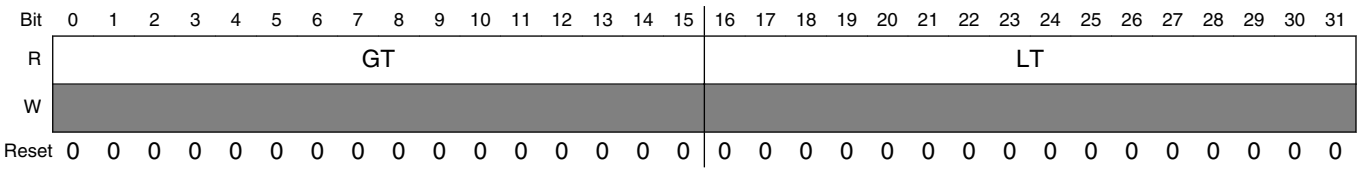
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																NAV															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_TTCAN_TURNA field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–31 NAV	Numerator Actual Value. <=0x0EFFF Illegal value 0x0F000-20FFF Actual numerator value >=0x21000 Illegal value

48.4.5.57 TT Local and Global Time register (M_TTCAN_TTLGT)

Address: 0h base + 134h offset = 134h

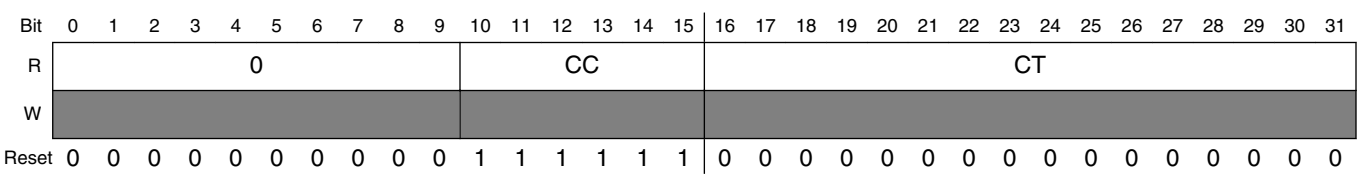


M_TTCAN_TTLGT field descriptions

Field	Description
0–15 GT	Global Time. Non-fractional part of the sum of the node's local time and its local offset (see Local time, cycle time, global time, and external clock synchronization). 0x0000-FFFF Global time value of M_TTCAN network
16–31 LT	Local Time. Non-fractional part of local time, incremented once each local NTU (see Local time, cycle time, global time, and external clock synchronization). 0x0000-FFFF Local time value of M_TTCAN node

48.4.5.58 TT Cycle Time and Count register (M_TTCAN_TTCTC)

Address: 0h base + 138h offset = 138h



M_TTCAN_TTCTC field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CC	Cycle Count. 0x00-3F Number of actual Basic Cycle in the System Matrix
16–31 CT	Cycle Time. Non-fractional part of the difference of the node's local time and Ref_Mark (see Local time, cycle time, global time, and external clock synchronization). 0x0000-FFFF Cycle time value of M_TTCAN Basic Cycle

48.4.5.59 TT Capture Time register (M_TTCAN_TTCPT)

Address: 0h base + 13Ch offset = 13Ch

Bit	0–15	16–25	26–31
R	SWV	0	CCV
W			
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0

M_TTCAN_TTCPT field descriptions

Field	Description
0–15 SWV	Stop Watch Value. On a rising/falling edge (as configured via TTOCN[SWP]) at the Stop Watch Trigger pin , when TTOCN[SWS] is not equal to "00" and TTIR[SWE] is '0', the actual time value as selected by TTOCN[SWS] (cycle, local, global) is copied to SWV and TTIR[SWE] will be set to '1'. Capturing of the next stop watch value is enabled by resetting TTIR[SWE]. 0x0000-FFFF Captured Stop Watch value
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 CCV	Cycle Count Value. Cycle count value captured together with SWV. 0x00-3F Captured cycle count value

48.4.5.60 TT Cycle Sync Mark register (M_TTCAN_TTCSM)

Address: 0h base + 140h offset = 140h

Bit	0–15	16–31
R	0	CSM
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

M_TTCAN_TTCSM field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CSM	Cycle Sync Mark. The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid. 0x0000-FFFF Captured cycle time

48.4.6 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_TTCAN module.

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_TTCAN module can be configured to allocate up to 1344 words in the Message RAM. It is not necessary to configure each of the sections listed in the following figure, nor is there any restriction with respect to the sequence of the sections.

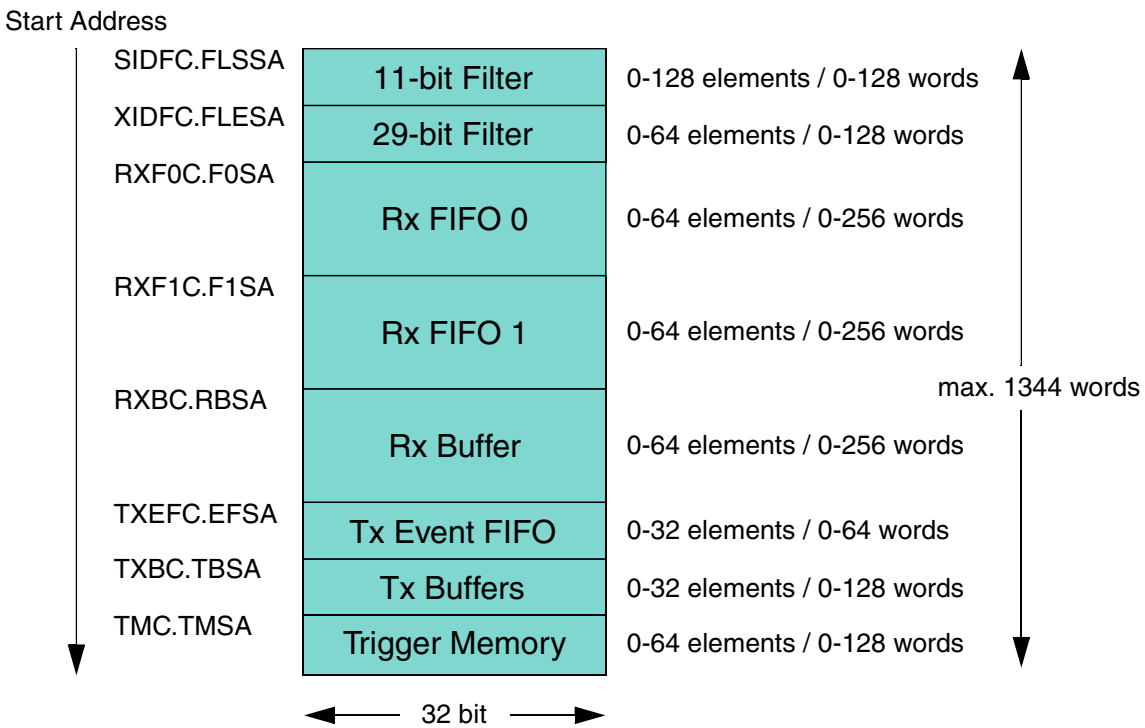


Figure 48-17. Message RAM Configuration

When the M_TTCAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

NOTE

The M_TTCAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

48.4.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure.

Table 48-17. Rx FIFO Element

0	7	8	15	16	23	24	31
R0	ESI	XTD	RTR	ID[28:0]			
R1	ANMF	FIDX[6:0]		res	EDL	BRS	DLC[3:0]	RXTS[15:0]
R2	DB3[7:0]			DB2[7:0]		DB1[7:0]	DB0[7:0]
R3	DB7[7:0]			DB6[7:0]		DB5[7:0]	DB4[7:0]

Table 48-18. Rx FIFO Element descriptions

R0 Bit 0 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 Bit 1 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier 0 11-bit standard identifier 1 29-bit extended identifier
R0 Bit 2 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame
R0 Bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 0 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE] 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 Bits 1:7 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 Bit 10 EDL Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 Bit 11 BRS Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 Bits 12:15 DLC[3:0]: Data Length Code	0-8 Received frame has 0-8 data bytes 9-15 Received frame has 8 data bytes

Table continues on the next page...

Table 48-18. Rx FIFO Element descriptions (continued)

R1 Bits 16:31 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 0:7	DB3[7:0]: Data Byte 3
R2 Bits 8:15	DB2[7:0]: Data Byte 2
R2 Bits 16:23	DB1[7:0]: Data Byte 1
R2 Bits 24:31	DB0[7:0]: Data Byte 0
R3 Bits 0:7	DB7[7:0]: Data Byte 7
R3 Bits 8:15	DB6[7:0]: Data Byte 6
R3 Bits 16:23	DB5[7:0]: Data Byte 5
R3 Bits 24:31	DB4[7:0]: Data Byte 4

48.4.6.2 Tx buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC.TFQS and TXBC.NDTB.

Table 48-19. Tx Buffer Element

	0			7	8			15	16			23	24			31
T0	res	XTD	RTR	ID[28:0]												
T1	MM[7:0]				EFC	res			DLC[3:0]				res			
T2	DB3[7:0]				DB2[7:0]				DB1[7:0]				DB0[7:0]			
T3	DB7[7:0]				DB6[7:0]				DB5[7:0]				DB4[7:0]			

Table 48-20. Tx Buffer Element description

T0 Bit 1 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 Bit 2 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame
T0 Bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table continues on the next page...

Table 48-20. Tx Buffer Element description (continued)

T1 Bits 0:7 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 8 EFC	Event FIFO Control 0 Don't store Tx events 1 Store Tx events
T1 Bits 12:15 DLC[3:0]	Data Length Code 0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes
T2 Bits 0:7	DB3[7:0]: Data Byte 3
T2 Bits 8:15	DB2[7:0]: Data Byte 2
T2 Bits 16:23	DB1[7:0]: Data Byte 1
T2 Bits 24:31	DB0[7:0]: Data Byte 0
T3 Bits 0:7	DB7[7:0]: Data Byte 7
T3 Bits 8:15	DB6[7:0]: Data Byte 6
T3 Bits 16:23	DB5[7:0]: Data Byte 5
T3 Bits 24:31	DB4[7:0]: Data Byte 4

48.4.6.3 Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

Table 48-21. Tx Event FIFO Element

0	7	8	15	16	23	24	31
E0	ESI | XTD | RTR	ID[28:0]					
E1	MM[7:0]	ET[1:0]	EDL | BRS	DLC[3:0]	TXTS[15:0]		

Table 48-22. Tx Event FIFO Element

E0 Bit 0 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 Bit 1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier

Table continues on the next page...

Table 48-22. Tx Event FIFO Element (continued)

E0 Bit 2 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 Bits 3:31 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 Bits 0:7 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status
E1 Bits 8:9ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 Bit 10 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 Bit 11 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 Bits 12:15 DLC[3:0]	Data Length Code 0-8 Frame with 0-8 data bytes transmitted 9-15 Frame with 8 data bytes transmitted
E1 Bits 16:31 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

48.4.6.4 Standard message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC.FLSSA plus the index of the filter element (0…127).

Table 48-23. Standard Message ID Filter Element

	0		7	8	15	16	23	24	31
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		res	SFID2[10:0]		

Table 48-24. Standard Message ID Filter Element Field Description

Bits 0:1 SFT[1:0]		Standard Filter Type First ID of standard ID filter element. 00 Range filter from SF1ID to SF2ID (SF2ID > SF1ID) 01 Dual ID filter for SF1ID or SF2ID 10 Classic filter: SF1ID = filter, SF2ID = mask 11 Reserved
Bit 2:4 SFEC[2:0]		Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag IE[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of SFT[1:0] ignored
Bits 5:15 SFID1[10:0]		Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a tandard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 21:31	SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: SFEC = ‘001’...‘110’ Second ID of standard ID filter element SFEC = ‘111’ Filter for Rx Buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B

Table continues on the next page...

Table 48-24. Standard Message ID Filter Element Field Description (continued)

		11 Debug Message C
	SFID2[8:6]	Is used to control the M_TTCAN filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_TTCAN host clock period in case the filter matches.
	SFID2[5:0]	Defines the offset to the Rx Buffer Start Address RXBC[RBSA] for storage of a matching message.

Note

In case a reserved value is configured, the filter element is considered disabled.

48.4.6.5 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0…63).

Table 48-25. Extended Message ID Filter Element

	0		7	8	15	16	23	24	31
F0	EFEC[2:0]			EFID1[28:0]					
F1	EFT[1:0]	res		EFID2[28:0]					

Table 48-26. Extended Message ID Filter Element Field Description

F0 Bits 0:2 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag IE[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches

Table continues on the next page...

Table 48-26. Extended Message ID Filter Element Field Description (continued)

		100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of EFT[1:0] ignored
F0 Bits 3:31 EFID1[28:0]		Extended Filter ID 1 First ID of extended ID filter element.
F1 Bits 0:1 EFT[1:0]		Extended Filter Type 00 Range filter from EF1ID to EF2ID (EF2ID ≤ EF1ID) 01 Dual ID filter for EF1ID or EF2ID 10 Classic filter: EF1ID = filter, EF2ID = mask 11 Range filter from EF1ID to EF2ID (EF2ID ≤ EF1ID), XIDAM mask not applied
F1 Bits 28:0	EFID2[28:0]	Extended Filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: EFEC = ‘001’...‘110’ Second ID of extended ID filter element EFEC = ‘111’ Filter for Rx Buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	EFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_TTCAN host clock period in case the filter matches.
	EFID2[5:0]	Defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

48.4.6.6 Trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address TTTMC[TMSA] plus the index of the trigger memory element (0…63).

Table 48-27. Trigger Memory Element

	0 7	8 15	16 23	24 31
T0	TM[15:0]		res / CC[6:0]	ASC[1:0] / TMIN / TMEX / TYPE[3:0]
T1	res	FTYPE / MNR[6:0]	res	MSC[2:0]

Table 48-28. Trigger Memory Element Field Description

T0 Bit 0:15 TM[15:0]	Time Mark Cycle time for which the trigger becomes active.
T0 Bit 17:23 CC[6:0]	Cycle Code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. 0b000000x valid for all cycles 0b000001c valid every 2nd cycle at cycle count mod2 = c 0b00001cc valid every 4th cycle at cycle count mod4 = cc 0b0001ccc valid every 8th cycle at cycle count mod8 = ccc 0b001cccc valid every 16th cycle at cycle count mod16 = cccc 0b01ccccc valid every 32nd cycle at cycle count mod32 = ccccc 0b1cccccc valid every 64th cycle at cycle count mod64 = cccccc
T0 Bit 24:25 ASC[1:0]	Asynchronous Serial Communication 00 No ASC operation 01 Reserved, do not use 10 Node is ASC receiver 11 Node is ASC transmitter
T0 Bit 26 TMIN	Extended Filter ID 2 Second ID of extended ID filter element.
T0 Bit 27 TMEX	Time Mark Event External 0 No action

Table continues on the next page...

Table 48-28. Trigger Memory Element Field Description (continued)

	1 Pulse at output m_ttcan_tmp with the length of one m_ttcan_cclk period is generated when the time mark of the trigger memory element becomes active and TTOCN.TTMIE = '1'
T0 Bit 28:31 TYP[3:0]	Trigger Type 0000 Tx_Ref_Trigger - valid when not in Gap 0001 Tx_Ref_Trigger_Gap - valid when in Gap 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window 0101 Tx_Trigger_Merged - starts a merged arbitration window 0110 Watch_Trigger - valid when not in Gap 0111 Watch_Trigger_Gap - valid when in Gap 1000 Rx_Trigger - check for reception 1001 Time_Base_Trigger - only control TMIN, TMEX, and ASC 1010…1111=End_of_List - illegal type, causes config error **Note:** For ASC operation (ASC = "10", "11") only trigger types Rx_Trigger and Time_Base_Trigger should be used.
T1 Bit 8 FTYPE	Filter Type 0 11-bit standard message ID 1 29-bit extended message ID
T1 Bit 9:15 MNR[6:0]	Message Number Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31. Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127
T1 Bits 29:31 MSC[2:0]	Message Status Count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). 0-7= Actual status Note: The trigger memory elements have to be written when the M_TTCAN is in INIT state.Write access to the trigger memory elements outside INIT state is not allowed.There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (TTOST[RTO]).

48.4.7 M_TTCAN functional description

48.4.7.1 Operating modes

48.4.7.2 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (=Bus_Idle) before it can take part in bus activities and start the message transfer

Access to the M_TTCAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write). CCCR[CCE] can only be set/ reset while CCCR[INIT] = '1'. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set:

- HPMS - High Priority Message Status
- RXF0S - Rx FIFO 0 Status
- RXF1S - Rx FIFO 1 Status
- TXFQS - Tx FIFO/Queue Status
- TXBRP - Tx Buffer Request Pending
- TXBTO - Tx Buffer Transmission Occurred
- TXBCF - Tx Buffer Cancellation Finished
- TXEFS - Tx Event FIFO Status
- TTOST - TT Operation Status
- TTLGT - TT Local and Global Time, only Global Time TTLGT[GT] is reset
- TTCTC - TT Cycle Time and Count
- TTCSM - TT Cycle Sync Mark

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writeable while CCCR[CCE] = ‘0’

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR[TEST] and CCCR[MON] can only be set by the Host while CCCR[INIT] = ‘1’ and CCCR[CCE] =‘1’. Both bits may be reset at any time. CCCR[DAR] can only be set/ reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

48.4.7.3 Normal operation

The M_TTCAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (TTOCF[OM] = "00"). It is required that both CCCR[INIT]and CCCR[CCE] are set before the TT Operation Mode can be changed.

Once the M_TTCAN is initialized and CCCR[INIT] is reset to zero, the M_TTCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

48.4.7.4 CAN FD Operations

There are two stages in the CAN FD protocol, first the Long Frame Mode where the data field of a CAN frame may be longer that 8 bytes. The second stage is the Fast Frame Mode where control field, data field, and CRC field of a CAN Frame are transmitted with a higher bit rate than the beginning and the end of the frame. Fast Frame Mode can only be used in combination with Long Frame Mode.

The CAN operation mode is chosen by programming CCCR.CME. In case CCCR.CME = “01” transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CCCR.CME = “10”/“11” transmission and reception of long and fast CAN FD frames is enabled. CCCR.CME can only be changed while CCCR.INIT and CCCR.CCE are both set.

When initialization is left (CCCR.INIT set to ‘0’), both Long Frame Mode and Fast Frame Mode are inactive, they have to be requested by writing to CCCR.CMR.

A mode change requested by writing to CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event CCCR.CMR is reset to “00” and the status flags CCCR.FDBS and CCCR.FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CCCR.CMR is retained until it is overwritten by the next mode change request. Default is normal CAN operation. A change of the CAN operation mode should not be requested while there are pending transmission requests.

When CCCR.CME ≠ “00”, received frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD long frame, EDL = dominant signifies a standard CAN frame. In a CAN FD long frame, the two bits following EDL, r0 and BRS, decide whether this CAN FD long frame is also a CAN FD fast frame. A CAN FD fast frame is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

The status bits CCCR.FDO and CCCR.FDBS indicate the format of transmitted frames. When CCCR.FDO is set, frames will be transmitted in CAN FD long format with EDL = recessive. When both CCCR.FDO and CCCR.FDBS are set, frames will be transmitted in CAN FD Fast format with both EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table.

Table 48-29. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing and Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing and Prescaler. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter’s error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is

transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter's error state. The data length code of CAN FD remote frames is transmitted as zero.

In case a M_TTCAN Tx Buffer is configured for CAN FD transmission with DLC > 8, the first 8 bytes are transmitted as configured in the Tx Buffer while the remaining part of the data field is padded with 0xCC. When the M_TTCAN receives a CAN FD frame with DLC > 8, the first 8 bytes of that frame are stored into the matching Rx Buffer or Rx FIFO. The remaining bytes are discarded.

48.4.7.4.1 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via M_TTCAN_Tx pin, the protocol controller receives the transmitted data from its local CAN transceiver via M_TTCAN_Rx pin. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.The following figure describes how the transceiver loop delay is measured

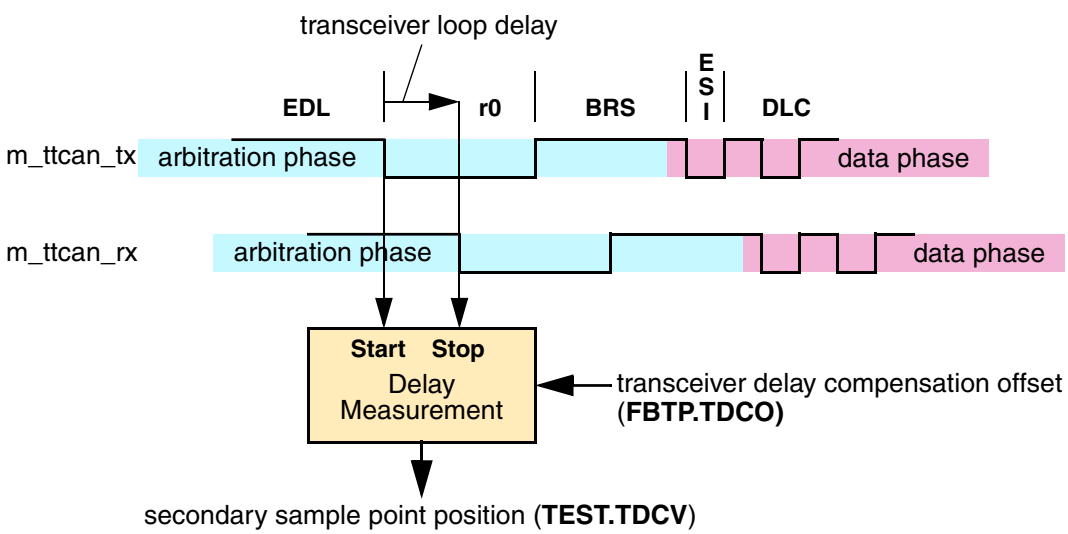


Figure 48-18. Transceiver delay measurement

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at M_TTCAN_Tx pin and the data received at M_TTCAN_Rx pin , starting with the falling edge of bit EDL. The delay is measured in CAN clock frequency periods.

A secondary sample point is calculated by adding a configurable transceiver delay compensation offset FBTP[TDCO] to the measured transceiver delay. The transceiver delay compensation offset is used to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta tq.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following sample point. During arbitration phase the delay compensation is always disabled.

The maximum delay which can be compensated by the M_TTCAN's delay compensation during the data phase is three bit times.

48.4.7.4.2 Configuration and status

Compensation for the transceiver loop delay by the M_TTCAN is enabled via FBTP[TDC]. The transceiver delay compensation offset is configured via FBTP[TDCO]. The actual delay compensation value applied by the M_CAN's protocol engine can be read from TEST[TDCV].

48.4.7.5 Bus Monitoring mode

The M_TTCAN is set in Bus Monitoring Mode by programming CCCR.MON to one or when error level S3 (TTOST[EL] = "1") is entered. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register TXBRP is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. The following figure shows the connection of signals M_TTCAN transmit output and M_TTCAN transmit output to the M_TTCAN in Bus Monitoring Mode.

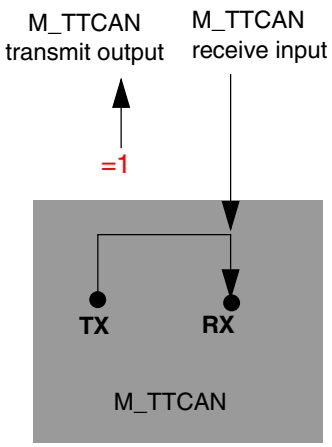


Figure 48-19. Pin Control in Bus Monitoring Mode

48.4.7.6 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

48.4.7.7 Frame transmission in DAR mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:

- Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
- Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

48.4.7.8 Power down (Sleep mode)

The M_TTCAN can be set into power down mode controlled by clock stop request input signal or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one.

When all pending transmission requests have completed, the M_TTCAN waits until bus idle state is detected. Then the M_TTCAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_TTCAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs M_TTCAN host clock and M_TTCAN clock may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting clock stop request signal respectively CC Control Register flag CCCR.CSR. The M_TTCAN will acknowledge this by resetting output clock stop acknowledge signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

48.4.7.9 Test modes

To enable write access to register TEST (see Test Register (M_TTCAN_TEST), bit CCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions. Four output functions are available for the M_TTCAN transmit pin by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_TTCAN's bit timing and it can drive constant dominant or recessive values. The actual value at M_TTCAN receive pin can be read from TEST.RX. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST.TX until the new configuration is visible at output pin. This applies also when reading input pinvia TEST.RX.

Note

Test modes should be used for production tests or self test only. The software control for M_TTCAN transmit pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

48.4.7.9.1 Watchdog mode

The application watchdog is served by reading register TTOST. When the application watchdog is not served in time, bit TTOST.AWE is set, all M_TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit TTOCF[AWL] to 0x00. The TT Application Watchdog should not be disabled in a M_TTCAN application program.

48.4.7.9.2 External Loop Back mode

The M_TTCAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_TTCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) an Rx Buffer or an Rx FIFO. Figure 48-20 shows the connection of transmit and receive signals to the M_TTCAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_TTCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back mode. In this mode the M_TTCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the input pin is disregarded by the M_TTCAN. The transmitted messages can be monitored at the transmit pin.

48.4.7.9.3 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST.LBCK and CCCR.MON to one. This mode can be used for a "Hot Selftest", meaning the M_TTCAN can be tested without affecting a running CAN system connected to the M_TTCAN Tx and Rx pins .

In this mode pin Rx pin is disconnected from the M_TTCAN and Tx pin is held recessive. The following figure shows the connection of M_TTCAN TX and _TTCAN Rx to the M_TTCAN in case of Internal Loop Back Mode.

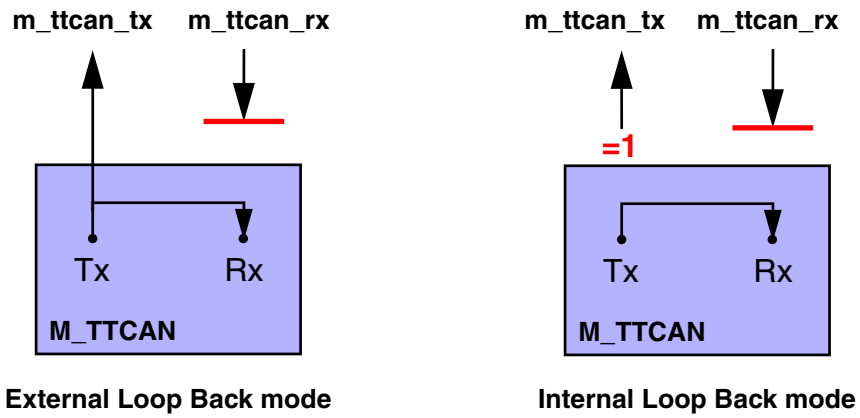


Figure 48-20. Pin Control in Loop Back modes

48.4.7.10 Timestamp generation

For timestamp generation the M_TTCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TCV]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer/ Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

48.4.7.11 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_TTCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC[TCP] as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC].

The Timeout Counter can only be started while CCCR[INIT] = '0'. It is stopped when CCCR[INIT] = '1', e.g. when the M_TTCAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOsFIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

48.4.7.12 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO’s Put and Get Indices.

48.4.7.12.1 Acceptance filtering

The M_TTCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering

- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Note

When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

48.4.7.12.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = "00": The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = "11": The Extended ID AND Mask (XIDAM) is not used for range filtering.

48.4.7.12.1.2 Filter for dedicated IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID=SF2ID resp. EF1ID=EF2ID.

48.4.7.12.1.3 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID /EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

48.4.7.12.1.4 Standard message ID filtering

The figure below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in Standard message ID Filter element.

Controlled by the Global Filter Configuration (GFC) and the Standard ID Filter Configuration (SIDFC) Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements

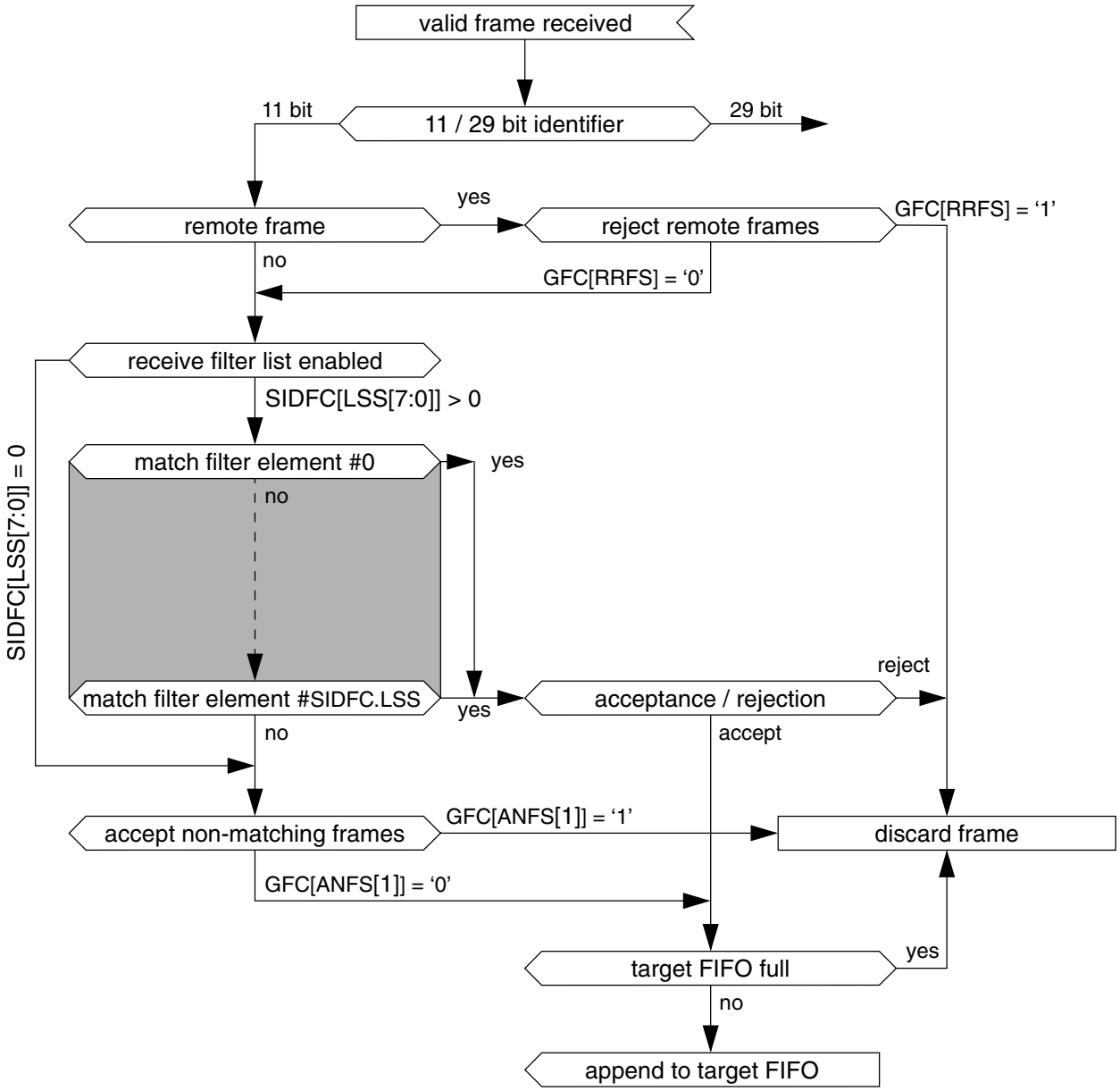


Figure 48-21. Standard Message ID Filter Path

48.4.7.12.1.5 Extended message ID filtering

The figure below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in Extended message ID filter element.

Controlled by the Global Filter Configuration GFC and the Extended ID Filter Configuration XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

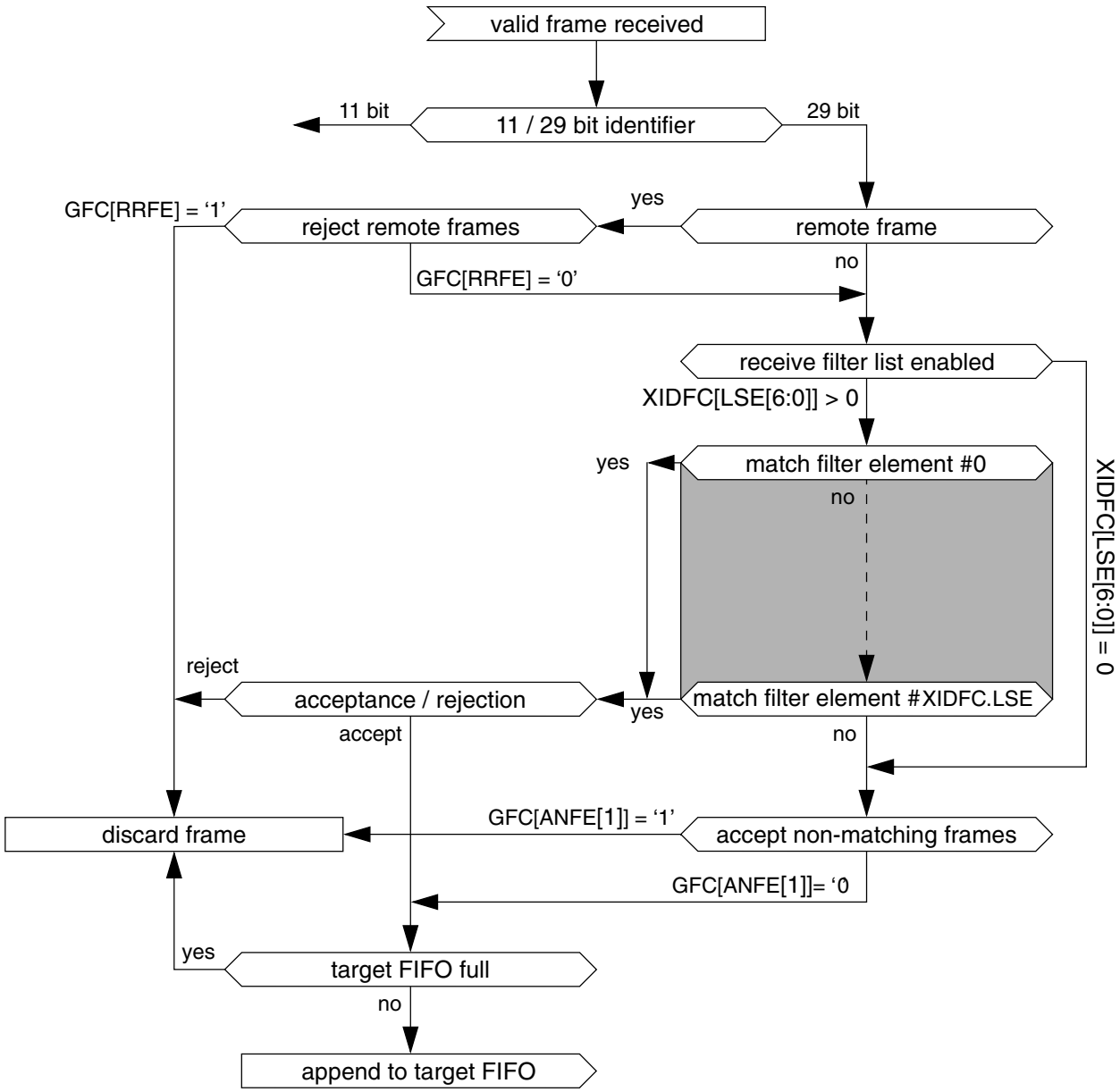


Figure 48-22. Extended Message ID Filter Path

48.4.7.13 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C. Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see Acceptance filtering. The Rx FIFO element is described in Rx

FIFOs. When an Rx FIFO full condition is signalled by IR[RFnF]], no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag IR[RFnL] is set.

To avoid an Rx FIFO over ow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC[FnWM], interrupt flag IR[RFnW] is set.

When reading from an Rx FIFO, four times the Rx FIFO Get Index RXFnS[FnGI] has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

48.4.7.14 Dedicated Rx Buffers

The M_TTCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured.

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 48-30. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration

48.4.7.14.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

48.4.7.15 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element.

Advantage: Fixed start address for the DMA transfers (relative to RXBC[RBSA]), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = “111” have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the M_TTCAN DMA request output is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while M_TTCAN DMA request output is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets M_TTCAN DMA acknowledge. This resets M_TTCAN DMA request output. Now the M_CAN is prepared to receive the next set of debug messages.

NOTE

To use full ‘Debug on CAN Support’ feature on a M_TTCAN instance, a DMA channel is required. Refer to device DMA map, to see which M_TTCAN instances has been assigned with DMA channel.

48.4.7.15.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to “111”. In this case fields

SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR[DRX] are set. The reception of debug messages can be monitored via RXF1S[DMS].

Table 48-31. Example Filter Configuration for Debug Messages

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

48.4.7.15.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S[DMS].

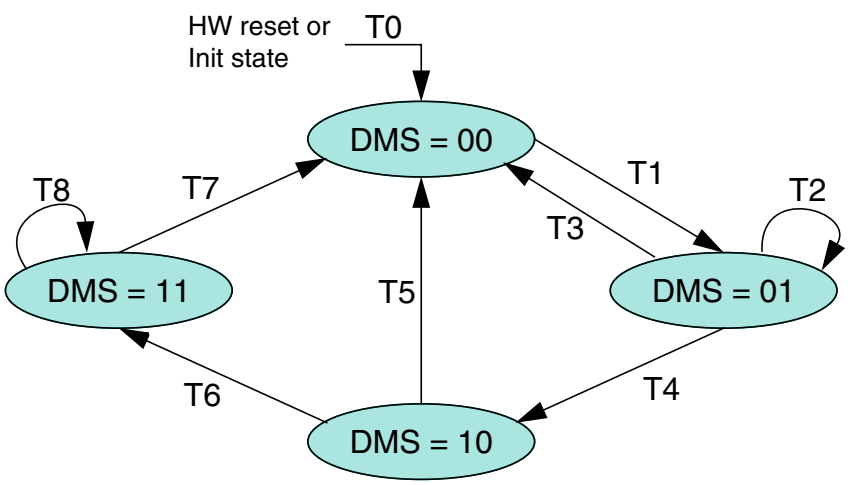


Figure 48-23. Debug Message Handling State Machine

T0: reset M_CAN DMA request output, enable reception of debug messages A, B, and C

T1: reception of debug message A

T2: reception of debug message A

T3: reception of debug message C

T4: reception of debug message B

T5: reception of debug messages A, B

T6: reception of debug message C

T7: DMA transfer completed

T8: reception of debug message A,B,C (message rejected)

48.4.7.16 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_TTCAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_TTCAN while TTM_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_TTCAN DMA ACK signal. This resets M_TTCAN DMA request signal. The debug message handling state machine enters idle state (DMS = "00") and waits for reception of the next debug messages.

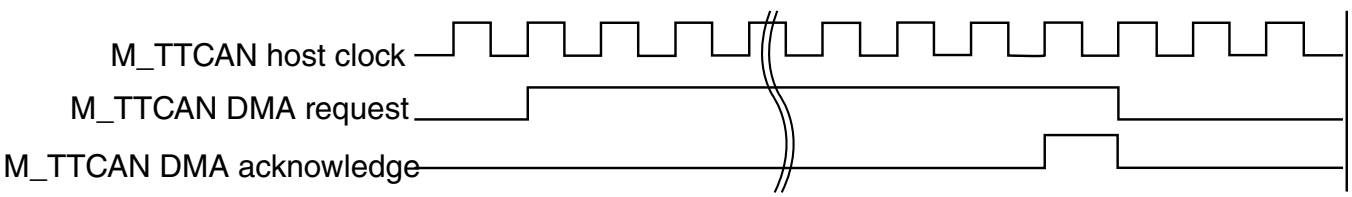


Figure 48-24. Timing of DMA interface signals

NOTE

If the DMA unit activates input signal M_TTCAN DMA ACK before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

48.4.7.17 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in Tx buffer element. The Tx Handler

starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

48.4.7.17.1 Dedicated Tx buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first. If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding four times the transmit buffer index (0…31) to the Tx Buffer Start Address TXBC[TBSA].

48.4.7.17.2 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_TTCAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements. New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF]= '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented. When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index. When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of

free Tx Buffers as indicated by the Tx FIFO Free Level. When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged. A Tx FIFO element allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0…31) to the Tx Buffer Start Address TXBC[TBSA].

48.4.7.17.3 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index

TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF]= '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx Queue Put Index TXFQS[TFQPI] (0…31) to the Tx Buffer Start Address TXBC[TBSA].

48.4.7.17.4 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case, TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

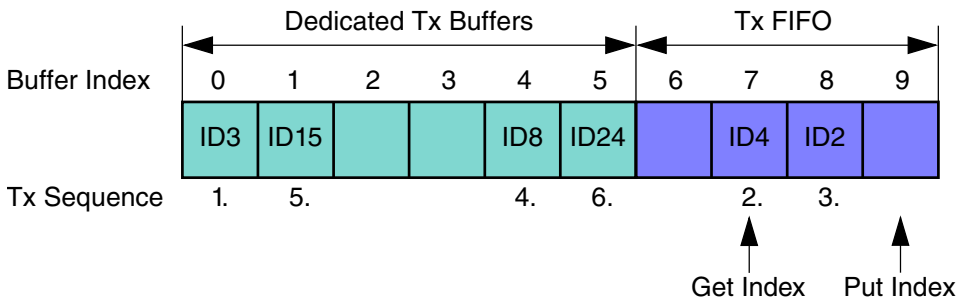


Figure 48-25. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

48.4.7.17.5 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only dedicated Tx Buffers are used.

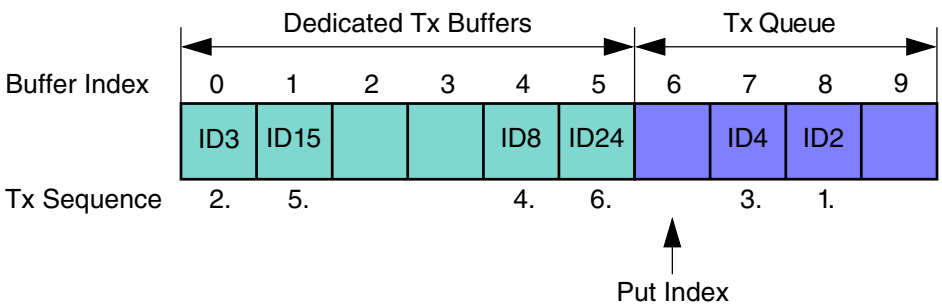


Figure 48-26. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

48.4.7.17.6 Transmit cancellation

The M_TTCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation. Successful cancellation is signalled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is star ed even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

48.4.7.17.7 Tx Event handling

To support Tx event handling the M_TTCAN has implemented a Tx Event FIFO. After the M_TTCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element. The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in Tx event FIFO element. When a Tx Event FIFO full condition is signalled by IR[TEFF], no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR[TEFL] is set.

To avoid a Tx Event FIFO over ow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR[TEFW] is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

48.4.8 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see Rx FIFO 0 Acknowledge Register (M_TTCAN_RXF0A), Rx FIFO 1 Acknowledge register (M_TTCAN_RXF1A) and Tx Event FIFO Acknowledge register (M_CAN_TXEFA), and) Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index. When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_TTCAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

48.4.9 M_TTCAN operations

48.4.9.1 Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message). For Level 1 the data length must be at least one; for Level 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference

message is configured via register TTRMC. The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted **Master_Ref_Mark** is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (**Next_is_Gap bit**) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters. The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

48.4.9.2 Level 1

Level 1 operation is configured via TTOCF[OM] = "01" and TTOCF[GEN]. External clock synchronization is not available in Level 1. The information related to the reference message is stored in the first data byte as shown below. Cycle_Count is optional.

Table 48-32. First byte of Level 1 reference message

Bits	***0***	***1***	***2***	3	***4***	***5***	6	7
First Byte	Next_is_Gap	res	Cycle_Count[5:0]					

48.4.9.3 Level 2

Level 2 operation is configured via TTOCF[OM] = "10" and TTOCF[GEN].The information related to the reference message is stored in the first four data bytes as shown below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Table 48-33. First four bytes of Level 2 reference message

Bits	0	***1***	2	3	***4***	***5***	6	7
First Byte	Next_is_Gap	res	Cycle_Count[5:0]					
Second Byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third Byte	Master_Ref_Mark[7:0]							

Table continues on the next page...

Table 48-33. First four bytes of Level 2 reference message (continued)

Fourth Byte	Master_Ref_Mark[15:8]

48.4.9.4 Level 0

Level 0 operation is configured via TTOCF[OM] = "11". External event-synchronized time-triggered operation is not available in Level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Table 48-34. First four bytes of Level 0 reference message

Bits	***0***	1	2	***3***	4	***5***	***6***	7
First Byte	Next_is_Gap	res	Cycle_Count[5:0]					
Second Byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third Byte	Master_Ref_Mark[7:0]							
Fourth Byte	Master_Ref_Mark[15:8]							

48.4.10 M_TTCAN configuration

48.4.10.1 M_TTCAN timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priority by the network system designer. In M_TTCAN Level 1 the NTU is the nominal CAN bit time. In M_TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable. In M_TTCAN Level 0 and Level 2 the length of the NTU is defined by the

Time Unit Ratio TUR. The TUR is in principle a non-integer number and given by the formula TUR = TURNA[NAV]/TURCF[DC]. The length of the NTU is given by the formula NTU = CAN Clock Period. TUR. The TUR Numerator Configuration NC is an 18-bit number, TURCF[NCL[15:0]] can be programmed in the range 0x0000-0xFFFF. TURCF[NCH[17:16]] is hard wired to 0b01. When the number 0xnnnn is written to TURCF[NCL[15:0]], TURNA[NAV] starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR Denominator Configuration TURCF[DC] is a 14-bit number. TURCF[DC] may be programmed in the range 0x0001 - 0x3FFF, 0x0000 is an illegal value.

In Level 1, NC must be ≤ 4 × TURCF[DC]. In Level 0, 2 NC must be ≤ 8 × TURCF[DC] to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets TURCF[DC] to 0x1000 and TURCF[NCL] to 0x10000, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the CCCR[INIT] is reset, or TURCF[ELT is set. TURCF[ELT] may not be set before the NTU is configured. Setting TURCF[ELT] to e1f also locks the write access to register TURCF.

At startup TURNA[NAV] is updated from NC (= TURCF[NCL] + 0x10000) when TURCF[ELT] is set. In M_TTCAN Level 1 there is no drift compensation. TURNA.NAV does not change during operation, it always equals NC.

In M_TTCAN Level 0 and Level 2 there are two possibilities for TURNA[NAV] to change. When operating as time slave or backup time master, and when TTOCF[ECC] is set, TURNA[NAV] is updated automatically to the value calculated from the monitored global time speed, as long as the M_TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master, and when TTOCF[EECS] is set, the Host may update TURCF[NCL]. When the Host sets TTOCN[ECS], TURNA[NAV] will be updated from the new value of NC at the next reference message. The status flag TTOST[WECS] as is set when TTOCN[ECS] is set and is cleared when TURNA[NAV] is updated. TURCF[NCL] is write locked while TTOST[WECS] is set.

In M_TTCAN Level 0 and Level 2 the clock calibration process adapts TURNA[NAV] in the range of the Synchronization Deviation Limit SDL of NC+/- $2^{(TTOCF[LDSDL]+5)}$. TURCF[NCL] should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of TURNA[NAV].

The synchronization deviation SD is the difference between NC and TURNA[NAV] (SD = |NC - TURNA[NAV]|). It is limited by the Synchronization Deviation Limit SDL, which is configured by its dual logarithm TTOCF[LDSDL] (SDL = $2^{(TTOCF[LDSDL]+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated TURNA[NAV] deviates by

more than SDL from NC, or if the Disc_Bit in the Reference Message is set, the drift compensation is suspended and TTIR[GTE] is set and TTOSC[QCS] is reset, or in case of the Disc_Bit = '1', TTIR[GTD] is set.

Table 48-35. TUR configuration examples

TUR	*8*	*10*	*24*	*50*	*510*	*125000*	*32.5*	*100/12*	*529/17*
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

TTOCN[ECS] schedules NC for activation by the next reference message. TTOCN[SGT] schedules TTGTP[TP] for activation by the next reference message. Setting of TTOCN[ECS] and TTOCN[SGT] requires TTOCF[EECS] to be set (external clock synchronization enabled) while the M_TTCAN is actual time master. The M_TTCAN module provides an application watchdog to verity the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit TTOCF[AWL] specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register TTOST. TTOST[AWE] indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag TTIR[AW] is set. For software development, the application watchdog may be disabled by programming TTOCF[AWL] to 0x00 (see also Watchdog mode).

48.4.10.2 Timing of interface signals

The timing events which cause a pulse at output M_TTCAN trigger time mark interrupt pulse and M_TTCAN register time mark interrupt pulse are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the Host clock domain (TTIR[TTMI] resp. TTIR[RTMI] set). The signals can be connected e.g. to the timing input(s) of another M_TTCAN node , in order to automatically synchronize two M_TTCAN networks. Output M_TTCAN start of cycle gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the Host clock domain.

48.4.11 M_TTCAN gap control

All functions related to Gap control apply only when the M_TTCAN is operated in external event synchronized time-triggered mode (**TTOCF[GEN]** = '1'). In this operation mode the M_TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit **Next_is_Gap** = '1' e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes **TTOCN[NIG**] = '1'. The **Next_is_Gap** bit will be transmitted as '1' with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin

by writing **TTOCN[GCS]** = '1'. When a reference message is started and **TTOCN[GCS]** is set, a HIGH level at event trigger pin will set **Next_is_Gap** = '1'.

As soon as that reference message is completed, the **TTOST[WFE]** bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, **TTOST[GSI]** will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit **TTOST[GSI]** will remain at '0'.

When a potential time master is in synchronization state In_Gap (**TTOST[SYS]** = "10"), it has four options to intentionally finish a Gap:

Under software control by writing **TTOCN[FGP]** = '1'.

Under hardware control (**TTOCN[GCS]** = '1') an edge from HIGH to LOW at the event-trigger input pin sets **TTOCN[FGP]** and restarts the schedule.

The third option is a time-triggered restart. When **TTOCN[TMG]** = '1', the next register time mark interrupt (**TTIR[RTMI]** = '1') will set **TTOCN[FGP]** and start the reference message.Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message. Setting of **TTOCN[FGP]** after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the

message schedule. When **TTOCN[FGP]** is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit **Next_is_Gap** = '1' in the reference message will be ignored, as well as the event-trigger input pinand the bits **TTOCN[NIG]**, **TTOCN[FGP]**, and **TTOCN[TMG]**.

48.4.12 Stop watch

The stop watch function enables capturing of M_TTCAN internal time values (local time, cycle time, or global time) triggered by an external event. To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via **TTOCN[SWS]**. When **TTOCN[SWS]** is ≠ '00' and TT Interrupt Register flag **TTIR[SWE]** is '0', the actual value of the time selected by **TTTOCN[SWS]** will be copied into **TTCPT[SWV]** on the next rising/falling edge (as configured via **TTOCN[SWP]**) on stop watch trigger pin . This will set interrupt flag **TTIR[SWE]**. After the application program has read **TTCPT[SWV]**, it may enable the next stop watch event by resetting **TTIR[SWE]** to '0'.

48.4.13 Local time, cycle time, global time, and external clock synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time. The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

Figure 48-27 describes the synchronization of the cycle time and global time, performed in the same manner by all M_TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame synchronization event. This frame synchronization event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark (Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register **TTCSM**. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

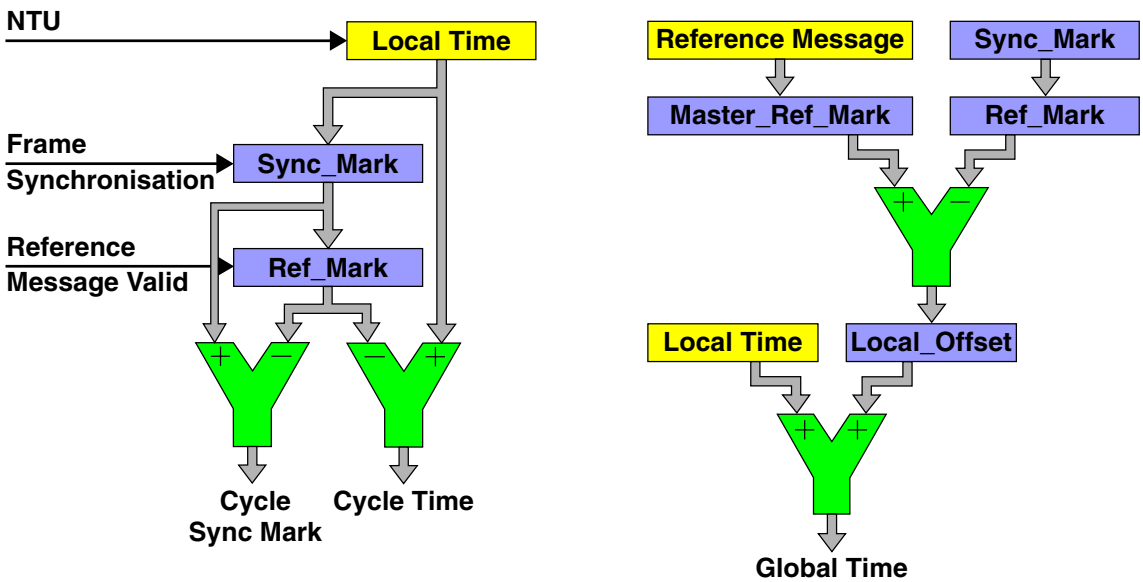


Figure 48-27. Cycle Time and Global Time Synchronization

The cycle time that can be read from **TTCTC[CT]** is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bit. The global time exists for M_TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3,4). The global time that can be read from **TTLGT[GT]** is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see in figure above). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signalled

by **Disc_Bit** in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register **TTLGT** is smoothed by a low-pass filtering to have a continuous monotonic value.

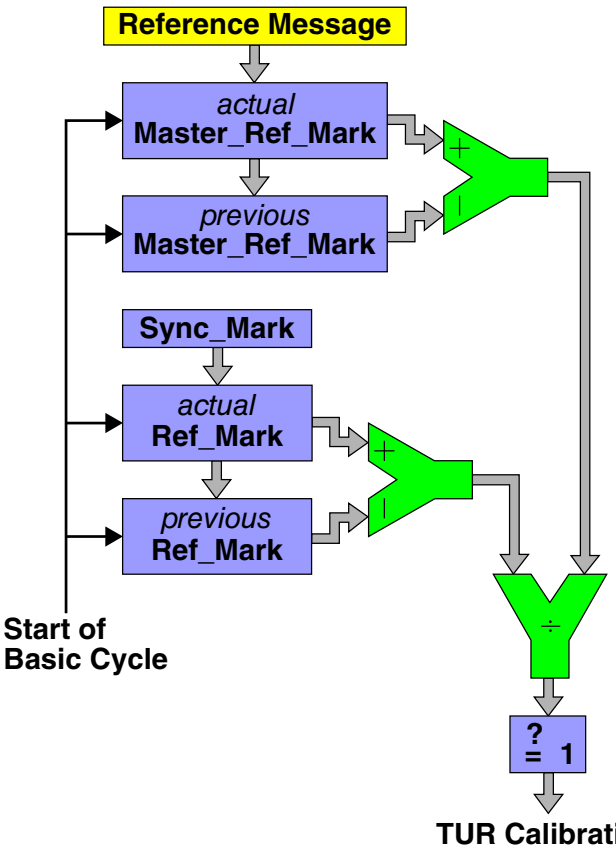


Figure 48-28. M_TTCAN Level 0 and Level 2 Drift Compensation

Figure 48-28 describes how in M_TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the **Disc_Bit** in the reference message is not set, a new value for **TURNA[NAV]** is calculated. If the Synchronization Deviation SD = |**NC - TURNA[NAV]**| ≤ SDL (Synchronization Deviation Limit), the new value for **TURNA[NAV]** takes effect. Else the automatic drift compensation is suspended.

In M_TTCAN Level 0 and Level 2, **TTOST[QCS]** indicates whether the automatic drift compensation is active or suspended. In M_TTCAN Level 1, **TTOST[QCS]** is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit **TTOCF[EECS]**.

The stop watch function (see Stop watch) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to **TURCF[NCL]** (**TURCF[DC]** cannot be updated during operation). The new value takes effect by writing **TTOCN[ECS]** to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register **TTGTP**. The new value takes effect by writing **TTOCN[SGT]** to '1'. The first reference message transmitted after the global time phase adjustment will have the **Disc_Bit** set to '1'.

TTOST[QGTP] shows whether the node's global time is in phase with the time master's global time. **TTOST[QGTP]** is permanently '0' in M_TTCAN Level 1 and when the Synchronization Deviation Limit is exceeded in M_TTCAN Level 0,2 (**TTOST[QCS]** = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a **Disc_Bit** = '1' or when **TTOST[QCS]**='0'.

48.4.14 M_TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

- S0 - No Error
- S1 - Warning- Only notification of application, reaction application-specific.
- S2 Error- Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset **TTOST[RTO]** set to the maximum value of 127.
- S3 - Severe Error

  Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and **CCCR[MON]** is set. The S3 error condition remains active until the application updates the configuration (set **CCCR[CCE]**).

  If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by **TTIR[ELC]**. The error level is monitored by **TTOST[EL]**.

The M_TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_Error (S3)

Sets Error Level **TTOST[EL]** to "11" when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.

- Watch_Trigger_Reached (S3)

  Sets Error Level **TTOST[EL]** to "11" when a watch trigger was reached because the reference message is missing.

- Application_Watchdog (S3)

  Sets Error Level **TTOST[EL]** to "11" when the application failed to serve the application watchdog.The application watchdog is configured via **TTOCF[AWL]**. It is served by reading register TTOST.When the watchdog is not served in time, bit **TTOST[AWE]** and interrupt flag **TTIR[AW]** are set, all M_TTCAN communication is stopped, and the M_TTCAN is set into bus monitoring mode (**CCCR[MON]**set to '1').

- CAN_Bus_Off (S3)

  Entering **CAN_Bus_Off** state sets error level **TTOST[EL]** to "11". **CAN_Bus_Off** state is signalled by **PSR[BO]** = '1' and **CCCR[INIT]** = '1'.

- Scheduling_Error_2 (S2)

  Sets Error Level **TTOST[EL]** to "10" if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag **TTIR[SE2]** is set. The Error Level **TTOST[EL]** is reset to "00" at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

- Tx_Overflow (S2)

  Sets Error Level **TTOST[EL]** to "10" when the Tx count is equal or higher than the expected number of Tx_T riggers **TTMLM[ENTT]** and a Tx_Trigger event occurs. In addition interrupt flag **TTIR[TXO]** is set. The Error Level **TTOST[EL]** is reset to "00" when the Tx count is no more than **TTMLM[ENTT]** at the start of a new matrix cycle.

- Scheduling_Error_1 (S1)

  Sets Error Level **TTOST[EL]** to "01" if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag **TTIR[SE1]** is set. If within one matrix cycle none of these conditions is valid, the Error Level **TTOST[EL]** is reset to "00".

- Tx_Underflow (S1)
- Sets Error Level **TTOST[EL]** to "01" when the Tx count is less than the expected number of Tx_Triggers **TTMLM[ENTT]** at the start of a new matrix cycle. In addition interrupt flag **TTIR[TXU]** is set. The Error Level **TTOST[EL]** is reset to "00" when the Tx count is at least **TTMLM[ENTT]** at the start of a new matrix cycle.

48.4.15 M_TTCAN message handling

48.4.15.1 Reference message

For potential time masters the identifier of the reference message is configured via **TTRMC[RID]**. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (M_TTCAN Level 1) resp. the first four data bytes (M_TTCANM_TTCAN Level 0 and Level 2) will be provided by the FSE. In case the Reference Message Payload Select **TTRMC[RMPS]** is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length **DLC** code from message buffer 0 is used.

Table 48-36. Number of Data Bytes transmitted with a Reference Messages

TTRMC.RMPS	TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4
1	0	4	1	4
1	1	4+MBO	1+MBO	4+MBO

To send additional payload with the reference message in Level 1 a **DLC** > 1 has to be configured, for Level 0, 2 a **DLC** > 4 is required. In addition the transmission request pending bit **TXBRP[TRP0]** of message buffer 0 must be set (see the table above). In case bit **TXBRP[TRP0]** is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier **TTRMC[RID]** is used.

48.4.15.2 Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see Rx handling).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

48.4.15.3 Message transmission

For time-triggered message transmission the M_TTCAN supplies 32 dedicated Tx buffers (see Dedicated Tx buffers). A Tx FIFO or Tx queue is not available when the M_TTCAN is configured for time-triggered operation (**TTOCF[OM]** = "01" or "10"). Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle. The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading **TXBTO**
- Update the Tx buffer's configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

Tx_Trigger_Continous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading **TXBCF**
- Update Tx buffer's configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

The message's **MSC** stored with the corresponding Tx_Trigger provides information on the success of the transmission. The **MSC** is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The **MSC** is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_TTCAN in Error Level S2 or Host has disabled this particular message). The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking **TXBRP**. If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading **TXBCF** before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

48.4.15.3.1 Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The **MSC** is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

48.4.15.3.2 Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The **MSC** is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

48.4.15.3.3 Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation. In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or cancelled by the Host), the four the request will be ignored. The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

48.4.16 M_TTCAN interrupt and error handling

The TT Interrupt Register **TTIR** consists off our segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register **TTIE**. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. The first segment consists of flags **CER**, **AW**, **WT**, and **IWT**.

Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of **IWT**, these error conditions require a re-configuration of the M_TTCAN module before the communication can be restarted.

The second segment consists of flags **ELC**, **SE1**, **SE2**, **TXO**, **TXU**, and **GTE**. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the M_TTCAN protocol's failure handling and do not require intervention by the application program.

The third segment consists of flags **GTD**, **GTW**, **SWE**, **TTMI**, and **RTMI**. The first two flags are controlled by global time events (Level 0,2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising edge on pininternal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by **TTOCN[TMC]** (Cycle, Local, or Global) equals time mark **TTTMK[TM]**. It can also be used to finish a Gap. The fourth segment consists of flags **SOG**, **CSM**, **SMC**, and **SBC**. These flags provide a means to synchronize the application program to the communication schedule.

48.4.17 Level 0

M_TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in M_TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1. Level 0 operation is configured via **TTOCF[OM]** = "11". In this mode the M_TTCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of **TTOCF[GEN]** is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages. In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (**TTIR[TTMI]**) is set when the cycle time has reached **TTOCF[IRTO]** × 0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (**TTIR[WT]**) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks. Register time mark interrupts (**TTIR[RTMI]**) are also possible. The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- **TTRMC**
- **TTOCF** except **EVTP**, **AWL**, **GEN**
- **TTMLM** except **ENTT**, **TXEW**
- **TURCF**

Level 0 operation is controlled via:

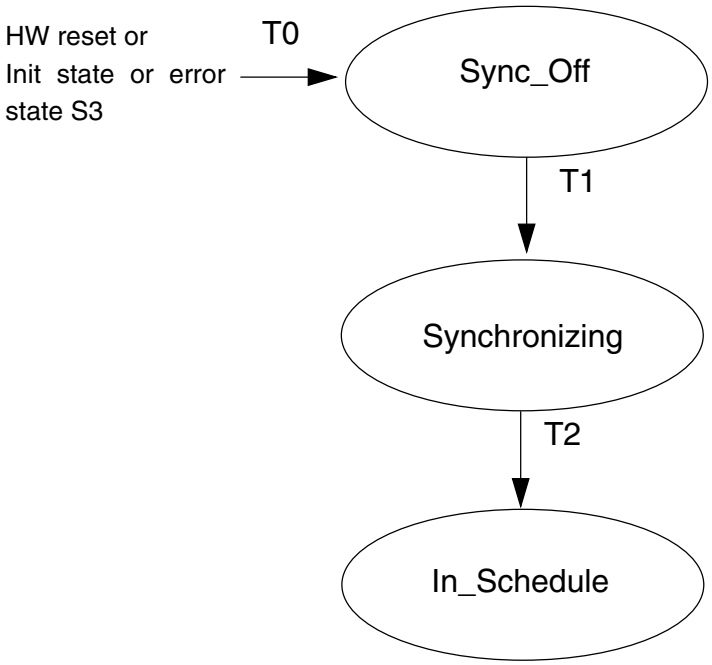
- **TTOCN** except **NIG**, **TMG**, **FGP**, **GCS**, **TTMIE**
- **TTGTP**
- **TTTMK**
- **TTIR** excluding bits **CER**, **AW**, **IWTSE2**, **SE1**, **TXO**, **TXU**, **SOG** (no function)
- **TTIR** the following bits have changed function
  - **TTMI** not defined by trigger memory - activated at cycle time **TTOCF[IRTO]** . 0x200
  - **WT** not defined by trigger memory - activated at cycle time 0xFF00

Level 0 operation is signalled via:

- TTOST excluding bits **AWE**, **WFE**, **GSI**, **GFI**, **RTO** (no function)

48.4.17.1 Synchronizing

The following figure describes the states and state transitions in M_TTCAN Level 0 operation. Level 0 has no In_Gap state.



T0: transition condition always taking prevalence

T1: Init state left, cycle time is zero

T2: at least two successive reference messages observed (last reference message did not contain a set Disc_Bit or Next_is_Gap bit)

Figure 48-29. Level 0 schedule synchronization state machine

48.4.17.2 Handling of error levels

During Level 0 operation only the following error conditions may occur:

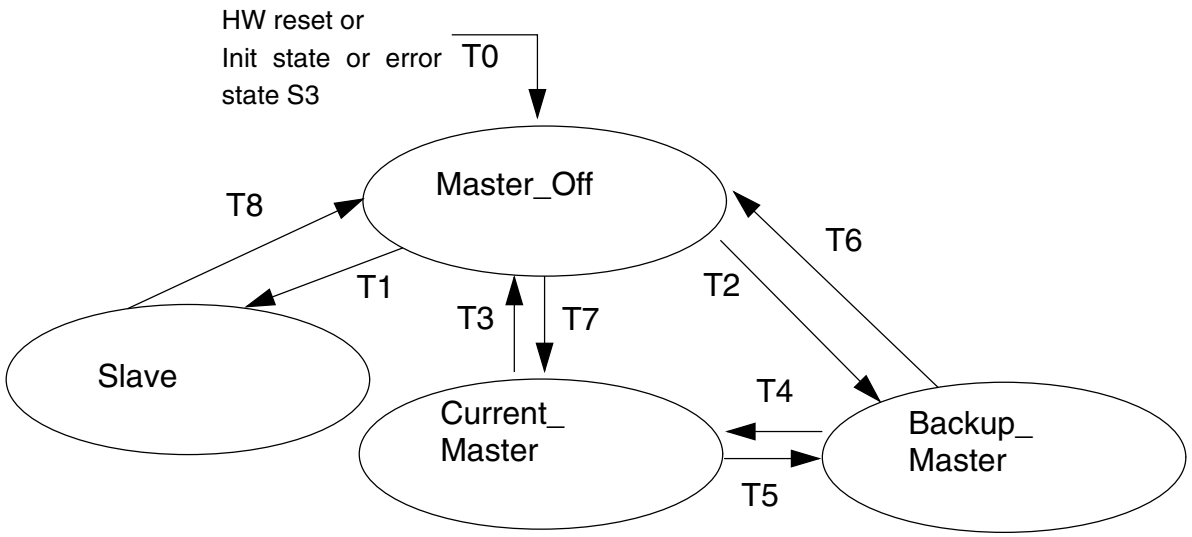
- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In M_TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both **TTOST[SYS**] and **TTOST[MS]** are reset, and interrupt flags **TTIR[GTE]** and **TTIR[GTD]** are set.

When error level S3 (**TTOST[EL]** = "11") is entered, bus monitoring mode is, contrary to M_TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

48.4.17.3 Master slave relation

The following figure describes the master slave relation in M_TTCAN Level 0. In case of an S3 error the M_TTCAN returns to state Master_Off.



T0: transition condition always taking prevalence

T1: reference message observed when not potential time master

T2: reference message observed with master priority ≠ own master priority, error state ≠ S3

T3: reference message observed with master priority =own master priority, error state ≠ S3

T4: reference message observed with own master priority

T5: reference message observed with master priority higher than own master priority

T6: error state S3

Figure 48-30. Level 0 master to slave relation

48.4.18 Synchronization to external time schedule

This feature can be used to synchronize the phase of the M_TTCAN's schedule to an external schedule (e.g. that of a second M_TTCAN network or FlexRay network). It is applicable only when the M_TTCAN is current time master (**TTOST[MS]** = "11"). External synchronization is controlled by event trigger input pin. If bit **TTOCN[ESCN]** is set, a rising edge at event trigger pin the M_TTCAN compares its actual cycle time with the target phase value configured by **TTGTP[CTP]**. Before setting **TTOCN[ESCN]** the Host has to adapt the phases of the two time schedules e.g. by using the M_TTCAN gap control (see M_TTCAN gap control). When the Host sets **TTOCN[ESCN]**, **TTOST[SPL]** is set.

If the difference between the cycle time and the target phase value **TTGTP[CTP]** at the rising edge at event trigger pin is greater than 9 NTU, the phase lock bit **TTOST[SPL]** is reset, and interrupt flag **TTIR[CSM]** is set. **TTOST[SPL]** is also reset (and **TTIR[CSM]** is set), when another node becomes time master. If both **TTOST[SPL]** and **TTOCN[ESCN]** are set, and if the difference between the cycle time and the target phase value **TTGTP[CTP]** at the rising edge at event trigger pin is less or equal 9 NTU, the phase lock bit **TTOST[SPL]** remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

Note

The rising edge detection at event trigger pin is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with TTGTP[CTP]. All further edges until the beginning of the next basic cycle are ignored.

48.5 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all masters and CPU,
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access.

Note

From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

48.5.1 Features

- Active low asynchronous reset
- 1–CPU + 5–CAN arbiter.

- Combined dynamic arbitration scheme: Bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth.
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access
- The SRAM access by a single master needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple masters are accessing the SRAM and it overlaps the address cycle of one master with the data cycle of another master. In case of multiple masters accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles.
- The peripheral GSI module enable acts as a request from each master. Each master will be made to wait at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth.
- Time slot is defined as one peripheral clock cycle.

48.5.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
  - CPU obtains every 2nd slot
  - CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
  - CPU obtains no time slots
  - CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains every 2nd time slot
  - Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):

- CPU obtains no time slots
- Active CAN 1/2 obtains every 2nd time slot

48.6 SRAM interface and memory organization

The CAN subsystem will interface with an external RAM using this interface. The RAM sizes required for the CAN subsystem are summarized in the following sections.

48.6.1 Shared memory

The following table/s shows maximum possible configuration per Filter, FIFO and Buffer but not all of these items can be configured to maximum size at the same time due to the smaller implemented physical memory as listed in Table 48-39.

Table 48-37. M_CAN message memory size requirement1.

Memory element	Number of elements	32-bit word per element	Total 32-bit words
11-bit Filter	128	1	128
29-bit Filter	64	2	128
Rx FIFO 0	64	18	1152
Rx FIFO 1	64	18	1152
Rx Buffer	64	18	1152
Tx Event FIFO	32	2	64
Tx Buffers	32	18	576
		M_CAN message memory size (word)	4352

1. M_CAN Rev 3.x supports CAN FD mode up to 64 data bytes

Table 48-38. M_TTCAN message memory size requirement1

Memory element	Number of elements	32-bit word per element	Total 32-bit words
11-bit Filter	128	1	128
29-bit Filter	64	2	128
Rx FIFO 0	64	4	256
Rx FIFO 1	64	4	256
Tx Event FIFO	32	2	64
Tx Buffers	32	4	128
Triggers	64	2	128
		M_TTCAN message memory size (word)	1344

1. M_TTCAN Rev 2.x supports CAN FD mode up to 8 data bytes.

Please refer Table 48-39 for the exact physical memory space used in the device.

48.6.2 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access. The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three bit error, a conventional SECDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2-bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

The ECC error address is reported at the top of the subsystem using a 16 bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

48.6.2.1 Features

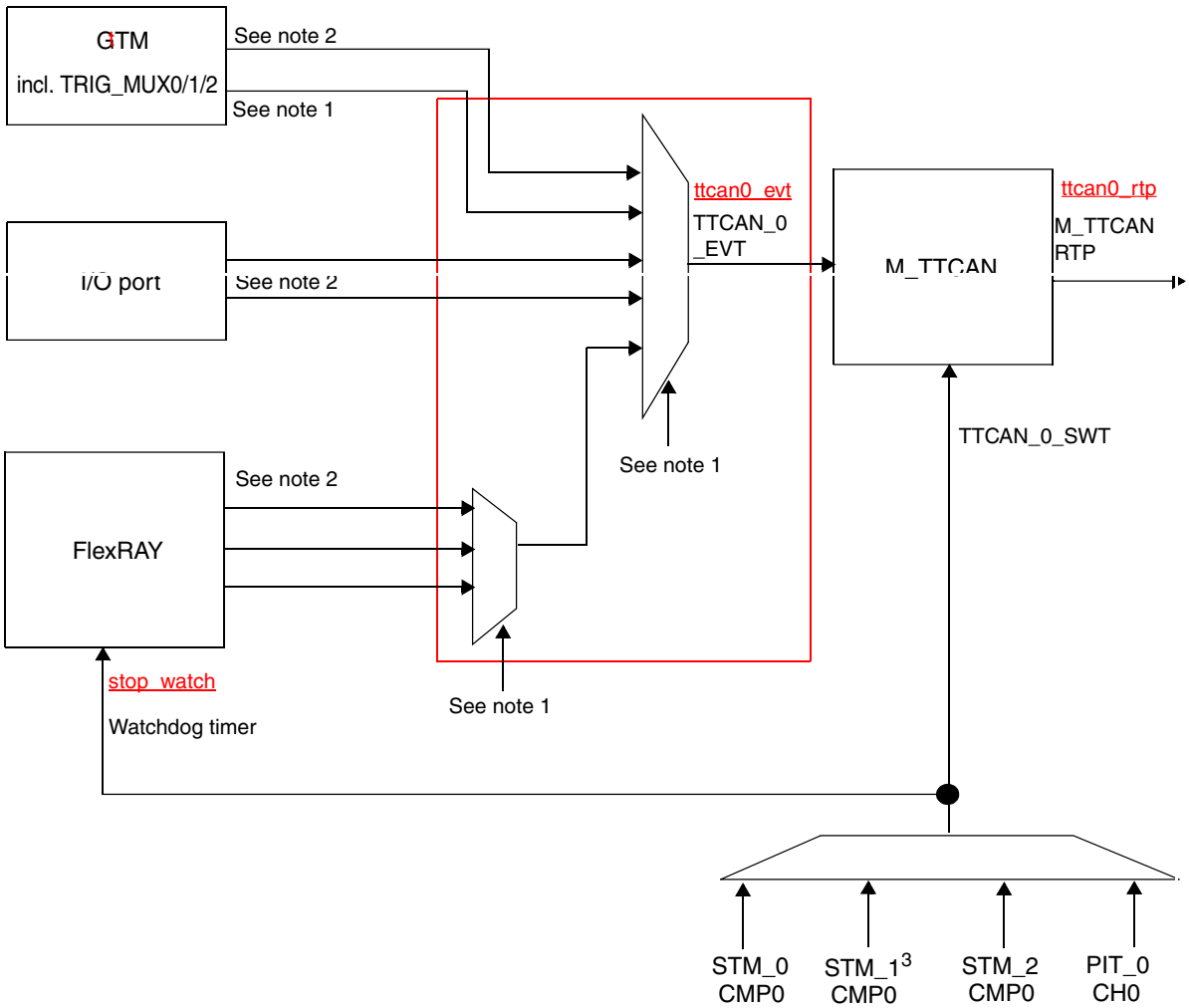
The ECC module supports the following:

- ecc_error = 00 ' No error is detected in the read data from memory
- ecc_error = 01 ' This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the master is error free.

- ecc_error = 10 ' This indicates that a two bit error has occurred within the codeword. In this case, no error correction is possible.
- ecc_error = 11 ' This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

48.7 Time Triggered CAN matrix and basic cycles management

The following figure shows hardware control for M_TTCAN cycles management.



Note 1: Multiplexers and control logic is implemented in the SIUL block. See SIUL Multiplexed Signal Configuration Register (SIUL2: MSCR 753, MSCR 754).
Note 2: For the signal details from various blocks see section SIUL Module-port MSCR assignments and SSS values (SIUL2: MSCR 753, MSCR 754).
Note 3: See device configuration chapter for the availability of the STM instances.

Figure 48-31. Hardware control for M_TTCAN cycles management

48.8 CAN nodes 1 and 2 I/Os sharing

Each CAN node will have its Tx/Rx signals connected to separated GPIO pairs from each others. It is required for MCAN_1 and MCAN_2 to share one common GPIO pair. This feature is described below.

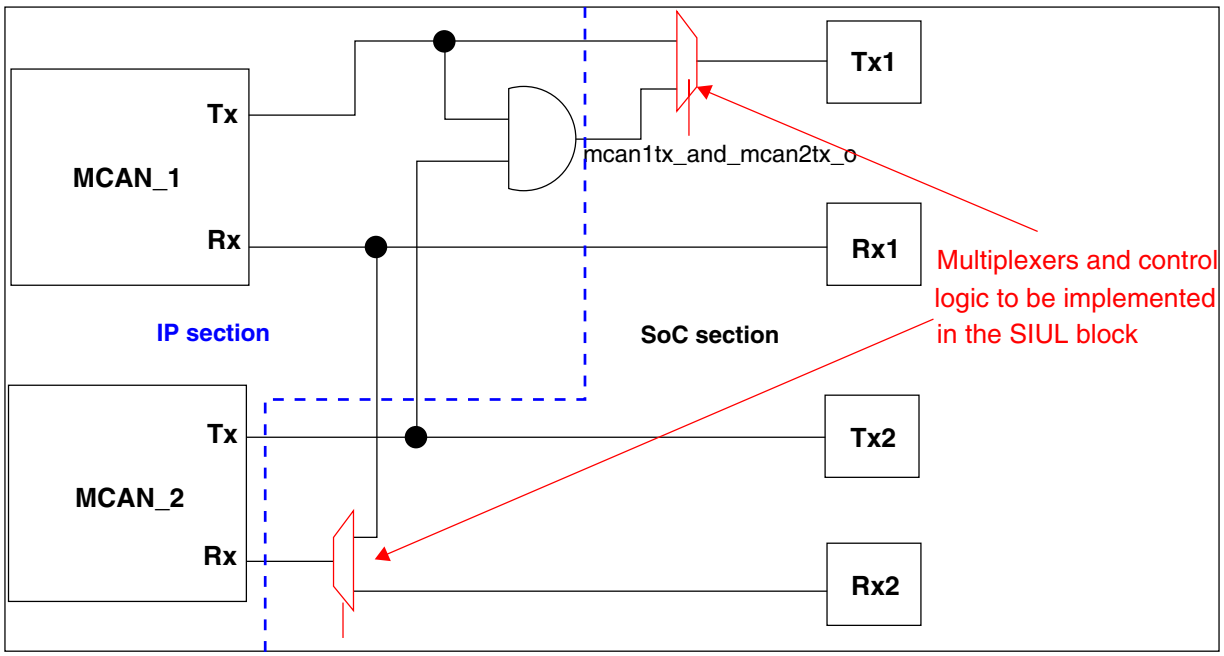


Figure 48-32. MCAN_1 and MCAN_2 connections to GP I/Os

48.9 External signal description

The CAN subsystem has following external pins.

- MCAN_x_RX: M_CAN x receive input
- MCAN_x_TX: M_CAN x transmit output
- M_TTCAN_x_RX: M_TTCAN x receive input
- M_TTCAN_x_TX: M_TTCAN x transmit output

48.10 Memory map and register description

48.10.1 Shared memory map

Table 48-39 shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 48-39. Shared memory map vs CAN transmit, receive and filters elements

Start Address Offset (Byte)	End Address Offset (Byte)	CAN block/sub block		Comment
0x0000	0x4FFF	M_CAN1 M_CAN2 M_CAN3 M_CAN4 M_TTCAN0	Standard Filters Extended Filters Rx FIFO0 Rx FIFO1 Rx Buffer Tx Event FIFO Tx Buffers M_TTCAN triggers	This available physical memory space could be configured for Filters/ FIFO/Buffer according to Table 48-37. This memory space can be flexible, assigned to any of the implemented M_CAN modules.

Note

There is no write access protection from a CAN module to the message memory space. The write access protection is only for CPU access to the message memory.

Note

The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

However the number of filters, Tx/Rx buffers, triggers used, it is mandatory for the user to configure the start address registers, specified in Table 48-40, in order to fit the memory mapping specified in Table 48-39.

Note

The CPU uses the peripheral interface to access the memory locations. It must be noted that there will not be any transfer error reported to the CPU in case of access to reserved memory locations.

Table 48-40. Detailed mapping in CAN subblocks

CAN sub block	Element name	Comment
Standard Filters	Standard Message ID filter 0 --- Standard Message ID filter 127	address configured in SIDFC [FLSSA]
Extended Filters	Extended Message ID filter 0	address configured in XIDFC [FLESA]

Table continues on the next page...

Table 48-40. Detailed mapping in CAN subblocks (continued)

CAN sub block	Element name	Comment
	--- Extended Message ID filter 63	
Rx FIFO0	Rx FIFO0 element 0 --- Rx FIFO0 element 63	address configured in RXF0C[F0SA]
Rx FIFO1	Rx FIFO1 element 0 --- Rx FIFO1 element 63	address configured in RXF1C[F1SA]
Rx Buffer	Rx Buffer element 0 --- Rx Buffer element 63	address configured in RXBC[RBSA
Tx Event FIFO	Tx event FIFO element 0 --- Tx event FIFO element 31	address configured in TXEFC[EFSA]
Tx Buffers	Tx Buffer element 0 --- Tx Buffer element 31	address configured in TXBC[TBSA]
M_TTCAN triggers (on M_TTCAN0 only)	Trigger element 0 --- Trigger element 63	address configured in TMC[TMSA]

Chapter 49
Deserial Serial Peripheral Interface (DSPI)

49.1 Introduction

The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note

For the chip-specific implementation details of the instances of this module, see the Device Configuration chapter.

49.1.1 Block diagram

The following figure provides the block diagram of this module.

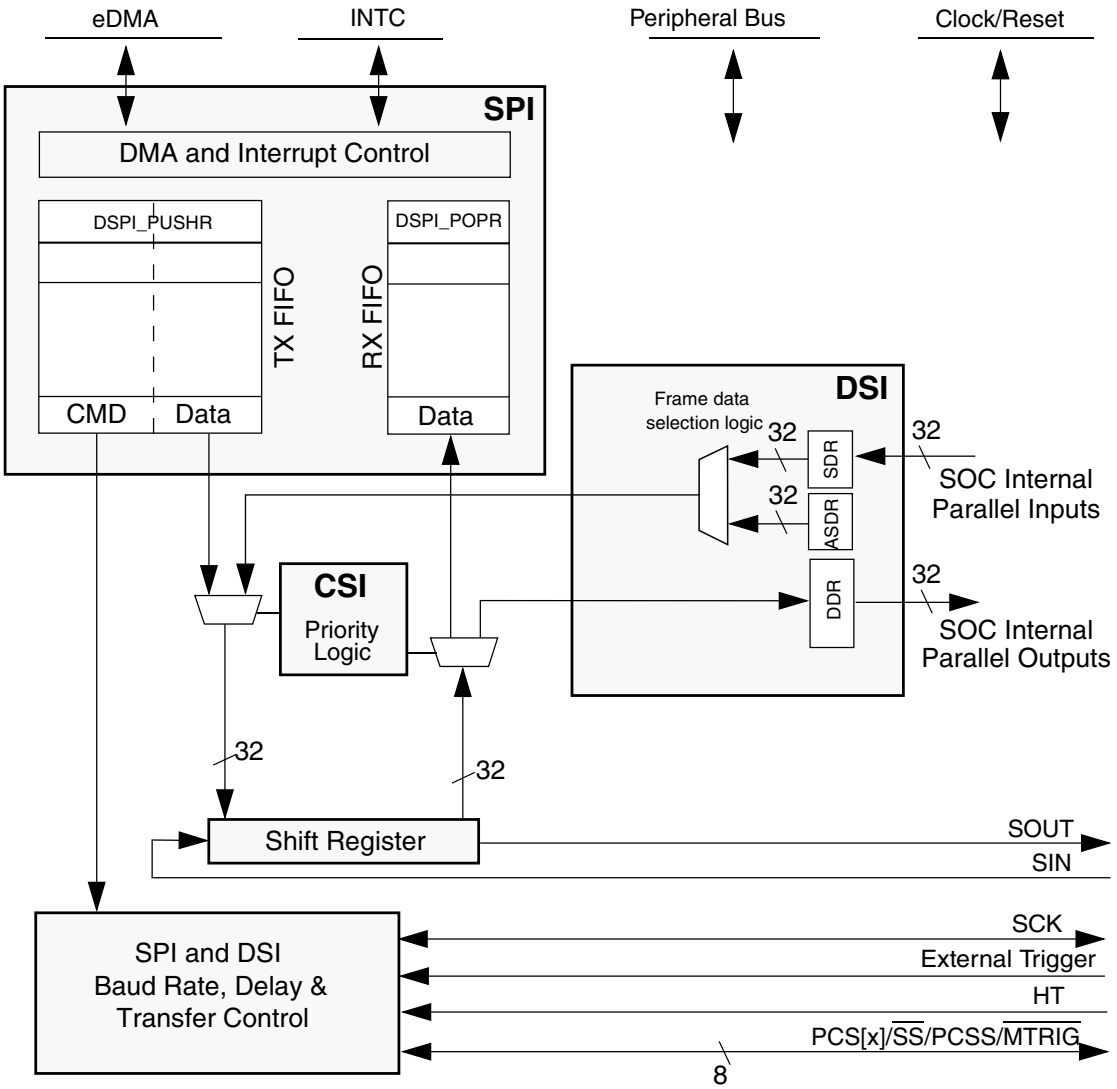


Figure 49-1. DSPI block diagram

49.1.2 Features

The DSPI supports the following features:

- Full-duplex, four-wire synchronous transfers.
- Master and Slave modes: data streaming operation in slave mode with continuous slave selection.
- Buffered transmit operation with 4-entry TX FIFO buffer.
- Buffered receive operation with 4-entry RX FIFO buffer.
- Asynchronous clocking scheme for Register and Protocol interfaces.

- Individual TX and RX FIFOs disabling for low-latency updates to SPI queues.
- Visibility inside TX and RX FIFOs for debugging.
- Programmable transfer attributes on a per-frame basis:
  - 8 transfer attribute registers along with 8 extended transfer attribute registers
  - serial clock with programmable polarity and phase
  - various programmable delays
  - programmable serial frame size of . SPI frames longer than 32 bits are supported using the continuous selection format
  - continuously held chip select capability
  - parity control
- 8 Peripheral Chip Selects, expandable to 256 with external demultiplexer.
- Deglitching support for up to 128 Peripheral Chip Selects with external demultiplexer.
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- Interrupt conditions:
  - End Of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame Complete (TCF)
  - Transfers due from current Command frame Complete (CMDTCF)
  - Transfer of current SPI frame Complete (SPITCF)
  - attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - frame received while Receive FIFO is full (RFOF)

    - SPI Parity Error (SPEF)
    - data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Global interrupt request line.
- Modified SPI transfer formats for communication with slower peripheral devices.
- Power-saving architectural features: support for stop mode.

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

- Two sources of serialized data:
    - DSPI memory-mapped register
    - parallel input signals
    - programmable selection of source data on bit basis.
- Deserialized data provided as:
    - parallel Output signals
    - bits in a memory-mapped register
- Interrupt conditions:
    - deserialized data matches pre-programmed pattern (DDIF)
    - transfer of current DSI frame complete (DSITCF)
    - DSI parity error (DPEF)
- DMA request support for following conditions:
    - deserialized data matches pre-programmed pattern (DDIF)
- Transfer initiation conditions:
    - continuous
    - edge sensitive hardware trigger
    - change in data
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics.
- Continuous serial communications clock.

DSPI supports a combination of SPI and DSI modes of operation (Combined Serial Interface (CSI)) for the downstream Micro Second Channel in either Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration (configurable by software). Both TSB and Interleaved TSB modes have the following common features:

- Transmission of frames is performed at frame boundaries.
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame.
- Separate interrupts for frame completion from SPI and DSI.

49.1.3 DSPI configurations

The DSPI module can operate in three configurations: SPI, DSI, and CSI. See the relevant section in the ‘Communication interfaces’ section of the ‘Device Configuration’ chapter.

49.1.3.1 SPI configuration

The SPI configuration allows the DSPI to send and receive serial data. This configuration allows the DSPI to operate as a basic SPI block with internal FIFOs supporting external queues operation. Transmit data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the receive FIFO and write transmit data to the transmit FIFO.

For queued operations, the SPI queues can reside in system RAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, DSPI and external queues in system RAM.

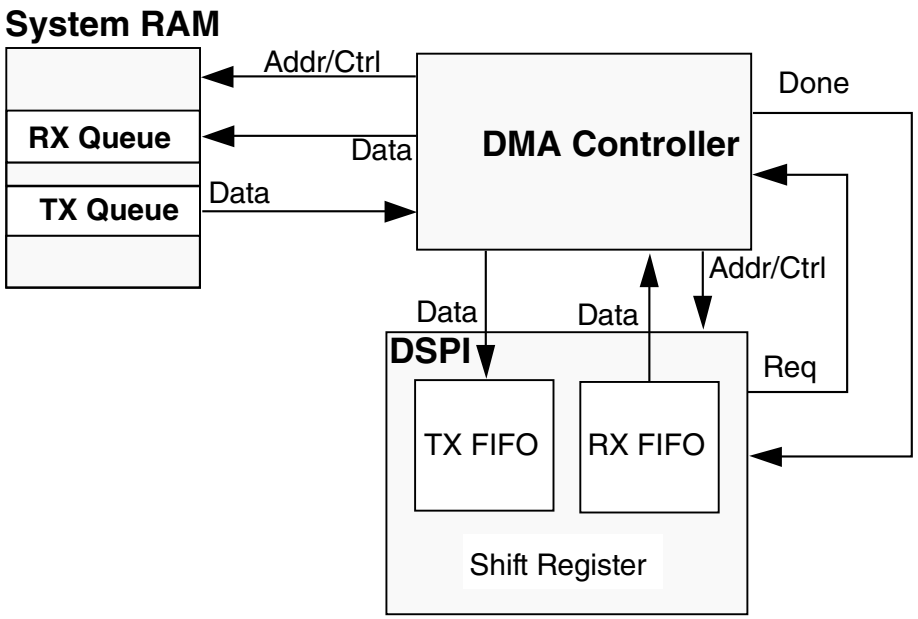


Figure 49-2. DSPI with queues and DMA

49.1.3.2 DSI configuration

In the Deserial Serial Interface (DSI) configuration, DSPI serializes up to 32 parallel input signals or register bits. . The DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. The data is transferred using a SPI-like protocol.

Timed Serial Bus (TSB) mode provides the MicroSecond Channel (MSC) downstream support, serializing from 4 to 32 parallel input signals or register bits. See Timed Serial Bus (TSB).

49.1.3.3 CSI configuration

There are three configurations available in this mode.

- The normal CSI configuration is a combination of the SPI and DSI configurations. In this configuration the DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.

- In the TSB configuration, transmission of SPI data has higher priority than DSI data.
- In the Interleaved TSB (ITSB) configuration, the frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. ITSB is detailed in Interleaved TSB (ITSB) Mode.

49.1.4 Modes of operation

The DSPI modes of operation that can be divided into two categories:

- Module-specific modes
  - Master mode
  - Slave mode
  - Module Disable mode
- MCU-specific modes
  - External Stop mode
  - Debug mode

The DSPI enters module-specific modes when the host writes a DSPI register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that an MCU may enter in parallel to the DSPI block-specific modes.

49.1.4.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK signal and the PCS[x] signals are controlled by the DSPI and configured as outputs.

49.1.4.2 Slave mode

Slave mode allows the DSPI to communicate with SPI/DSI bus masters In this mode, the DSPI responds to externally controlled serial transfers. The SCK signal and the PCS[0]/SS signals are configured as inputs and driven by a SPI bus master.

49.1.4.3 Module Disable mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory-mapped logic in the DSPI can be stopped while in Module Disable mode.

49.1.4.4 External Stop mode

External Stop mode is used for MCU power management; the DSPI supports the Peripheral Bus Stop mode mechanism.

Upon a request to enter External Stop mode:

- In Master mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary, it signals that the protocol clock to the DSPI module may be shut off.
- In Slave mode, a request to enter External Stop mode must be run after the last clock cycle or Low-power mode is not entered.

49.1.4.5 Debug mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls DSPI behavior in the debug mode. If the bit is set, the DSPI stops all serial transfers, when the MCU is in Debug mode. If the bit is cleared, the MCU Debug mode has no effect on the DSPI.

49.2 DSPI signal description

This section provides the DSPI signals description.

The following table lists the signals that may connect off chip depending on device implementation.

Table 49-1. DSPI signal description

Signal	Description	I/O
PCS0/ SS	Master mode: Peripheral Chip Select 0 output Slave mode: Slave Select input	I/O
PCS[3:1]	Master mode: Peripheral Chip Select 1–3	O

Table continues on the next page...

Table 49-1. DSPI signal description (continued)

Signal	Description	I/O
	Slave mode: Unused	
PCS4	Master mode: Peripheral Chip Select 4	O
PCS5/ PCSS[1]	Master mode: Peripheral Chip Select 5 / Peripheral Chip Select Strobe Slave mode: Unused	O
PCS[7:6]	Master mode: Peripheral Chip Select 6–7 Slave mode: Unused	O
SIN	Serial Data In	I
SOUT	Serial Data Out	O
SCK	Master mode: Serial Clock (output) Slave mode: Serial Clock (input)	I/O
HT	Hardware Trigger	I

1. This pin is not available for DSPI_2

49.2.1 PCS0/$\overline{SS}$ — Peripheral Chip Select/Slave Select

In master mode, the PCS0 signal is a Peripheral Chip Select output that selects which slave device the current transmission is intended for.

In slave mode, the active low SS signal is a Slave Select input signal that allows a SPI master to select the DSPI as the target for transmission.

49.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are Peripheral Chip Select output signals in master mode.

In slave mode, these signals are unused.

49.2.3 PCS4 — Peripheral Chip Select 4

In master mode, PCS4 is a Peripheral Chip Select output signal.

49.2.4 PCS5/$\overline{\text{PCSS}}$ — Peripheral Chip Select 5/Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the DSPI is in master mode and the MCR[PCSSE] bit is cleared, this signal selects which slave device the current transfer is intended for.

When the DSPI is in master mode and the MCR[PCSSE] bit is set, the $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4and PCS6 – PCS7 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in slave mode.

49.2.5 PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7

PCS[6] – PCS[7] are Peripheral Chip Select output signals in master mode. In slave mode, these signals are not used.

49.2.6 SIN — Serial Input

SIN is a serial data input signal.

49.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

49.2.8 SCK — Serial Clock

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

49.2.9 HT — Hardware Trigger

In master mode while in DSI or CSI Configurations, the HT signal initiates a data transfer when the DSICR0[TRRE] bit is set and a rising or falling edge is detected on HT. Which edge to trigger on is determined by the DSICR0[TPOL] bit.

49.3 Memory map/register definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the DSPIx_POPR also results in a transfer error.

DSPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	DSPI Module Configuration Register (DSPI_MCR)	32	R/W	0000_4001h	49.3.1/2087
8	DSPI Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	49.3.2/2090
C	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	49.3.3/2091
C	DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR_SLAVE0)	32	R/W	7800_0000h	49.3.4/2096
10	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	49.3.3/2091
10	DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR_SLAVE1)	32	R/W	7800_0000h	49.3.4/2096
14	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	49.3.3/2091
18	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	49.3.3/2091
1C	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	49.3.3/2091
20	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	49.3.3/2091
24	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR6)	32	R/W	7800_0000h	49.3.3/2091
28	DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR7)	32	R/W	7800_0000h	49.3.3/2091
2C	DSPI Status Register (DSPI_SR)	32	R/W	0201_0000h	49.3.5/2098
30	DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	49.3.6/2102
34	DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	49.3.7/2104
34	DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	49.3.8/2107
38	DSPI POP FIFO Register (DSPI_POPR)	32	R	0000_0000h	49.3.9/2108
3C	DSPI Transmit FIFO Registers (DSPI_TXFR0)	32	R	0000_0000h	49.3.10/ 2108
40	DSPI Transmit FIFO Registers (DSPI_TXFR1)	32	R	0000_0000h	49.3.10/ 2108
44	DSPI Transmit FIFO Registers (DSPI_TXFR2)	32	R	0000_0000h	49.3.10/ 2108

Table continues on the next page...

DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
48	DSPI Transmit FIFO Registers (DSPI_TXFR3)	32	R	0000_0000h	49.3.10/ 2108
7C	DSPI Receive FIFO Registers (DSPI_RXFR0)	32	R	0000_0000h	49.3.11/ 2109
80	DSPI Receive FIFO Registers (DSPI_RXFR1)	32	R	0000_0000h	49.3.11/ 2109
84	DSPI Receive FIFO Registers (DSPI_RXFR2)	32	R	0000_0000h	49.3.11/ 2109
88	DSPI Receive FIFO Registers (DSPI_RXFR3)	32	R	0000_0000h	49.3.11/ 2109
BC	DSPI DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	49.3.12/ 2109
C0	DSPI DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	49.3.13/ 2112
C4	DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	49.3.14/ 2112
C8	DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	49.3.15/ 2113
CC	DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)	32	R	0000_0000h	49.3.16/ 2113
D0	DSPI DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	49.3.17/ 2113
D4	DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	49.3.18/ 2115
E8	DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)	32	R/W	0000_0000h	49.3.19/ 2116
EC	DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)	32	R/W	0000_0000h	49.3.20/ 2116
11C	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE0)	32	R/W	0000_0001h	49.3.21/ 2117
120	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE1)	32	R/W	0000_0001h	49.3.21/ 2117
124	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE2)	32	R/W	0000_0001h	49.3.21/ 2117
128	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE3)	32	R/W	0000_0001h	49.3.21/ 2117
12C	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE4)	32	R/W	0000_0001h	49.3.21/ 2117
130	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE5)	32	R/W	0000_0001h	49.3.21/ 2117
134	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE6)	32	R/W	0000_0001h	49.3.21/ 2117
138	DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE7)	32	R/W	0000_0001h	49.3.21/ 2117

Table continues on the next page...

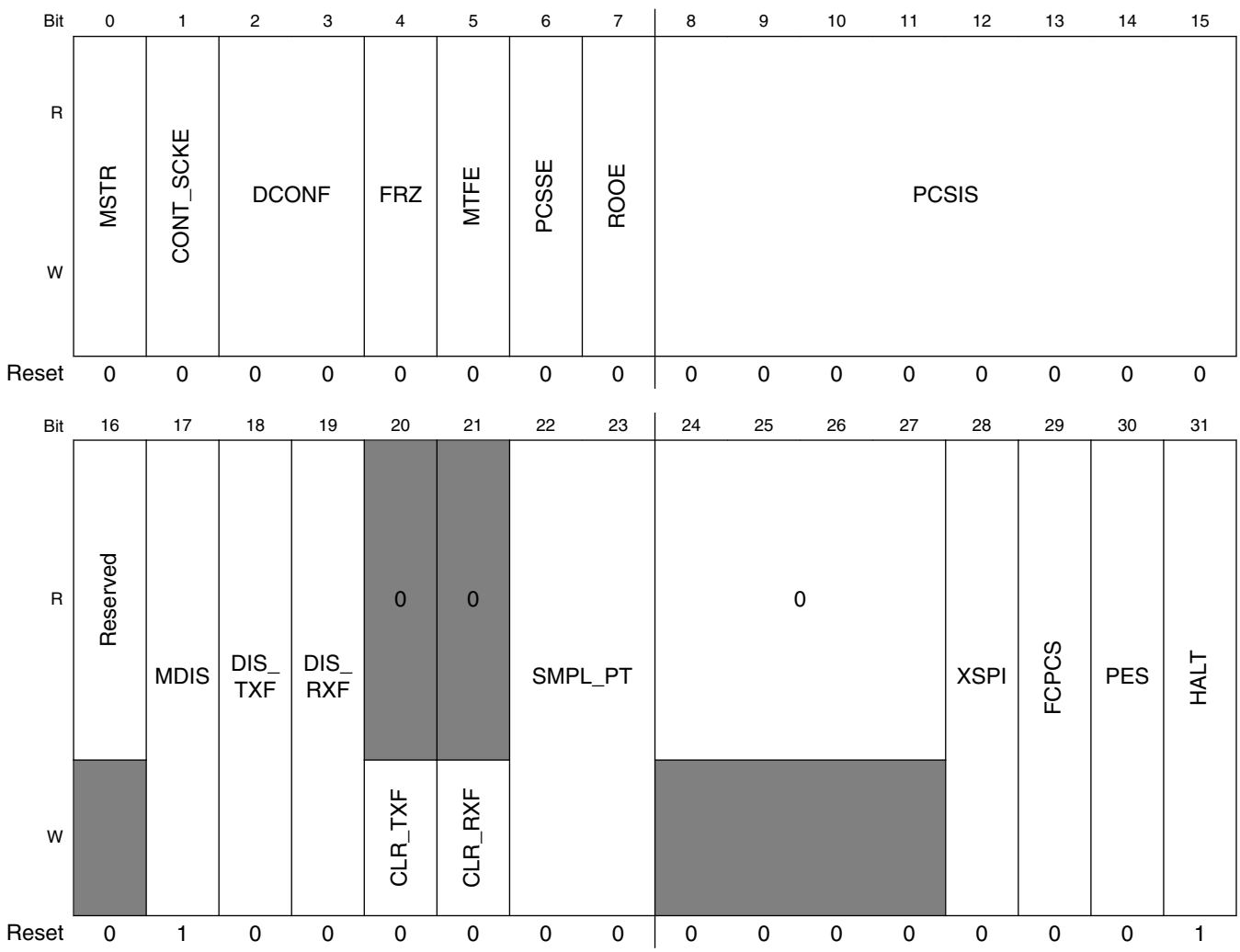
DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13C	DSPI Status Register Extended (DSPI_SREX)	32	R	0000_0000h	49.3.22/ 2118

49.3.1 DSPI Module Configuration Register (DSPI_MCR)

The DSPI_MCR contains bits to configure various attributes associated with DSPI operations. The HALT and MDIS bits can be changed at any time, but they only take effect on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed while the DSPI is in the Running state.

Address: 0h base + 0h offset = 0h



DSPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode. 1 DSPI is in master mode.
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	DSPI Configuration Selects the DSPI configuration. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in debug mode. 1 Halt serial transfers in debug mode.
5 MTFE	Modified Timing Format Enable Enables a modified transfer format to be used. MTFE should not be set in ITSB Mode as Microsecond Channel upstream operation is not supported. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS[5]/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS[5]/PCSS is used as the Peripheral Chip Select[5] signal (PCS[5]). 1 PCS[5]/PCSS is used as an active-low PCS Strobe signal (PCSS).
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the DSPI to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer generating the overflow is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only bit field is reserved and always has the value zero.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory-mapped logic in the DSPI, putting the DSPI in a software controlled power-saving state. The reset value of the MDIS bit is 1. MDIS should be set to ’0’ in Slave Mode as a slave doesn’t control master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (TXFIFO and CMD FIFO) operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Tx FIFO is enabled. 1 Tx FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Rx FIFO is enabled. 1 Rx FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes TX FIFO and CMD FIFO. Writing a '1' to CLR_TXF clears the TX FIFO and CMD FIFO Counters. The CLR_TXF bit is always read as zero. 0 Do not clear the Tx FIFO and CMD FIFO counters. 1 Clears the Tx FIFO and CMD FIFO counters.
21 CLR_RXF	Clear RX FIFO Flushes RX FIFO. Writing a '1' to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the Rx FIFO counter. 1 Clear the Rx FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the DSPI master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTAR is 0. 00 0 protocol clocks between SCK edge and SIN sample 01 1 protocol clock between SCK edge and SIN sample 10 2 protocol clocks between SCK edge and SIN sample 11 Reserved
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

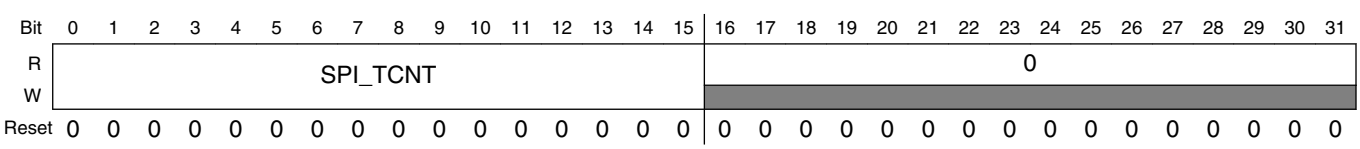
Field	Description
28 XSPI	Extended SPI Mode Enables use of CTARE (Command and Transfer Attribute Register Extended) Registers, which allow the user to send up to 32-bit SPI frames. Command Cycling is also enabled, allowing the user to send multiple Data Frames in a single Command Frame. When MCR[DIS_TXF] is asserted, Extended SPI Mode cannot be used to transmit SPI frames over 16 bits in size. 0 Normal SPI Mode. Up to 16-bit Frames. Command Cycling is not available. 1 Extended SPI Mode. Up to 32-bit SPI Frames. Command Cycling Enabled
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of "After SCK (t_{ASC})" and "PCS to SCK (t_{CSC})" delays when operating in Continuous PCS mode. The individual delay masks are selected via PUSHR[PE_MASC] and PUSHR{PP_MCSC]. The firmware should select appropriate masks when providing continuous frames via the PUSHR. This masking is not available if Continuous SCK mode is enabled. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt Starts and stops DSPI transfers. 0 Start transfers. 1 Stop transfers.

49.3.2 DSPI Transfer Count Register (DSPI_TCR)

DSPIx_TCR has an SPI transfer counter to assist in queue management.

Do not write the TCR when the DSPI is in the Running state.

Address: 0h base + 8h offset = 8h



DSPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter The SPI_TCNT field increments every time the last bit of a SPI frame is transmitted. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. A value written to SPI_TCNT presets the counter to that value. The Transfer Counter wraps around; incrementing past 65535 resets the counter to zero.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.3.3 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

CTARs are used to define different transfer attributes:

- In master mode, the CTARs define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays.
- In slave mode, a subset of the bitfields in CTAR0 and CTAR1 set the slave transfer attributes.

Do not write to the CTARs while the DSPI is in the Running state.

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Table 49-2. DSPI SCK duty cycle

DBR	CPHA	PBR	SCK duty cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 49-3. Delay scaler encoding

Field value	Delay scaler value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256
1000	512
1001	1024
1010	2048
1011	4096
1100	8192
1101	16384
1110	32768
1111	65536

Table 49-4. DSPI baud rate scaler

CTARn[BR]	Baud rate scaler value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32

Table continues on the next page...

Table 49-4. DSPI baud rate scaler (continued)

CTARn[BR]	Baud rate scaler value
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

Address: 0h base + Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate. (Master mode only) Doubles the baud rate of the Serial Communications Clock (SCK). It halves the Baud Rate division ratio, supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When DBR is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in Table 49-2. See the BR field description for details on how to compute the baud rate. Refer to Table 49-2 for DSPI SCK duty cycle values. 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. The minimum valid value for the number of bits to be transmitted for any mode is 4.

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	There is a constraint on the minimum allowable Frame size, which depends on the Register Read/Write clock to the Protocol Clock Ratio. The minimum Frame size (FMSZ + 1) is decided by the following equation. Upper Ceiling must be applied for non-integer values. Min. Frame Size = ((4 x fp) + (3 x fr)) / (n x fr) fp = Protocol Clock Frequency fr = Register Read/Write Clock Frequency n = Multiple of protocol clock required to get Baud Clock - Given by (PBR x BR) / (1 + DBR) The minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Note that 'f_p' can be equal to, less than or greater than 'f_r' purely based on user requirement. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register.
5 CPOL	Clock Polarity. (Master and Slave mode) Selects the inactive state of the Serial Communications Clock (SCK). Devices must have identical clock polarities for successful communication between serial devices. When the Continuous Selection Format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, this bit is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First. Specifies whether the LSB or MSB of the frame is transferred first. In TSB mode, this bit should comply with the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler. Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for calculating the PCS to SCK Delay. Refer to PCS to SCK delay (t_{CSC}) for more details. In TSB mode, PCSSCK has no effect. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3.

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10–11 PASC	After SCK Delay Prescaler. Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for calculating the After SCK Delay. Refer to After SCK delay (t_{ASC}) for more details. In TSB mode, PASC has no effect. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12–13 PDT	Delay after Transfer Prescaler. (Master mode only) Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In TSB mode, the PDT field defines two MSB bits of the Delay after Transfer. See the DT field description for calculating the Delay after Transfer. Refer to Delay after transfer (t_{DT}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
14–15 PBR	Baud Rate Prescaler. (Master mode only) Selects the prescaler value for the baud rate. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for calculating the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16–19 CSSCK	PCS to SCK Delay Scaler. (Master mode only) Selects the scaler value for the PCS to SCK delay. PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period calculated by: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$ Table 49-3 lists the delay scaler values. Refer to PCS to SCK delay (t_{CSC}) for more details. In TSB mode, the field has no effect.
20–23 ASC	After SCK Delay Scaler. (Master mode only) Selects the scaler value for the After SCK Delay.

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period calculated by: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer to After SCK delay (t_{ASC}) for more details.
24–27 DT	Delay After Transfer Scaler (Master mode only) Selects the Delay after Transfer Scaler. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period. The Delay after Transfer is a multiple of the protocol clock period calculated by: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer to PCS to SCK delay (t_{CSC}) for more details.
28–31 BR	Baud Rate Scaler. (Master mode only) Selects the scaler value for the baud rate. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is calculated by: SCK baud rate = $(f_P/PBR) \times [(1+DBR)/BR]$ Table 49-4 lists the baud rate scaler values.

49.3.4 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR_SLAVE*n*)

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

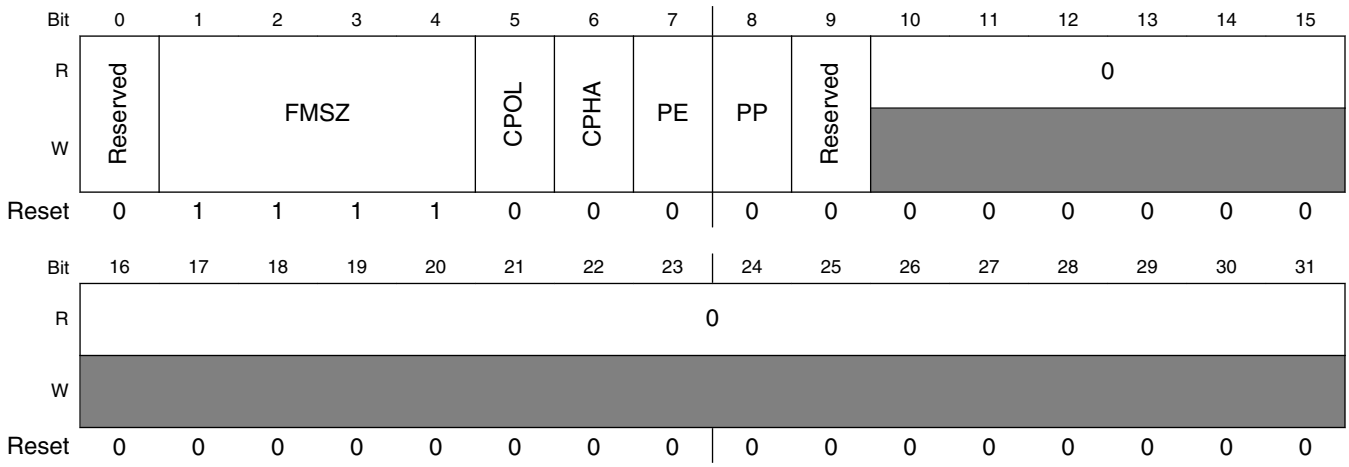
- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: 0h base + Ch offset + (4d × i), where i=0d to 1d



DSPI_CTAR_SLAVE*n* field descriptions

Field	Description
0 Reserved	This field is reserved. Always write the reset value to this field.
1–4 FMSZ	Frame Size - The number of bits transferred per frame is equal to the FMSZ field value plus 1. Note that the minimum valid value of frame size is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame.

Table continues on the next page...

DSPI_CTAR_SLAVE*n* field descriptions (continued)

Field	Description
	0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 Reserved	This field is reserved.
10–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

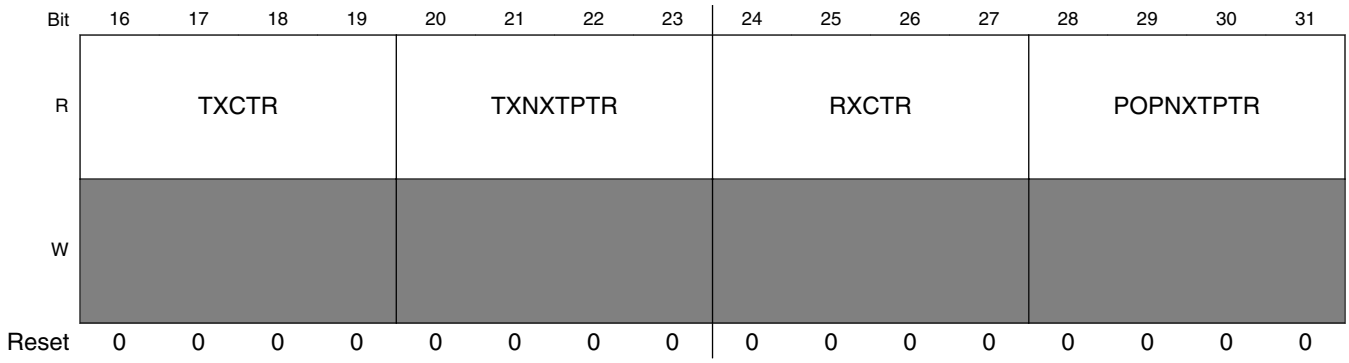
49.3.5 DSPI Status Register (DSPI_SR)

The status register contains DSPI status bits and interrupt/DMA request event flag bits.

Software can clear most of the flag bits in the SR by writing a ‘1’ to them (w1c). Writing a 0 to a flag bit has no effect. This register may not be writable in module disable mode due to the use of power saving mechanisms.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFIWF	RFDF	CMDFFF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1



DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag. Indicates that all bits in a frame have been shifted out. 0 Transfer not complete. 1 Transfer complete.
1 TXRXS	TX and RX Status. Reflects the run status of the DSPI. 0 Transmit and receive operations are disabled (DSPI is in stopped state). 1 Transmit and receive operations are enabled (DSPI is in running state).
2 SPITCF	SPI Frame Transfer Complete Flag. Similar to TCF but only asserted on SPI frame transmission completion in CSI Mode This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
3 EOQF	End of Queue Flag. Indicates that the last entry in a queue has been transmitted when the DSPI is in master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
4 TFUF	Transmit FIFO Underflow Flag. Indicates an underflow condition in the TX FIFO. Transmit underflow is detected only for DSPI blocks in slave mode and SPI configuration. TFUF is set when the TX FIFO of a DSPI operating in SPI slave mode is empty and an external SPI master initiates a transfer. 0 No Tx FIFO underflow. 1 Tx FIFO underflow has occurred.
5 DSITCF	DSI Frame Transfer Complete Flag. Similar to TCF but only asserted on DSI frame transmission completion in CSI Mode.

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
	This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
6 TFFF	Transmit FIFO Fill Flag. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing '1' to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 Tx FIFO is full. 1 Tx FIFO is not full.
7 BSYF	Busy Flag. This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmitting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. See Command First In First Out (CMD FIFO) buffering mechanism.. This bit is valid only when DSPI_MCR[XSPI] is enabled. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for current cyclic command transfer.
8 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. CMDTCF remains set until it is cleared by writing a '1' to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command complete.
9 DPEF	DSI Parity Error Flag. Indicates that a DSI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.
10 SPEF	SPI Parity Error Flag. Indicates that a SPI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits. Indicates that a DSI frame has been received with bits selected by DIMR with active polarity, defined by the DPIR. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
12 RFOF	Receive FIFO Overflow Flag. Indicates an overflow condition in the RX FIFO. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 TFIWF	Transmit FIFO Invalid Write Flag. Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. 0 No Invalid Data present in TX FIFO 1 Invalid Data present in TX FIFO since CMD FIFO is empty
14 RFDF	Receive FIFO Drain Flag. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 Rx FIFO is empty. 1 Rx FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag. Provides a method for the DSPI to request more entries to be added to the CMD FIFO. The CMDFFF bit is set while the CMD FIFO is not full. The CMDFFF is cleared by writing a ‘1’ to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16–19 TXCTR	TX FIFO Counter. Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the data part of PUSHR is written. The TXCTR is decremented every time the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer. Indicates which TX FIFO Entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter. Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer. Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

49.3.6 DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

The DSPI_RSER controls DMA and interrupt requests.

Do not write to the RSER while the DSPI is in the Running state.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TCF_RE	CMDFFF_RE	SPITCF_RE	EOQF_RE	TFUF_RE	DSITCF_RE	TFFF_RE	TFFF_DIRS	CMDTCF_RE	DPEF_RE	SPEF_RE	DDIF_RE	RFOF_RE	TFIWF_RE	RFDF_RE	RFDF_DIRS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFFF_DIRS	DDIF_DIRS	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable. Enables TCF flag in the SR to generate an interrupt request. 0 Disabled. 1 Enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the Status register to generate a request. DSPIx_RSER[CMDFFF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. Enables SPITCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
3 EOQF_RE	DSPI Finished Request Enable Enables the EOQF flag in the status register to generate an interrupt request.

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	0 Disabled. 1 Enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. Enables DSITCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the status register to generate a request. DSPIx_RSER[TFFF_DIRS] toggles between either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. DSPIx_SR[TFFF] flag and DSPIx_RSER[TFFF_RE] must be set. 0 Interrupt requests. 1 DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the DDIF flag in the status register to generate an interrupt requests. 0 Disabled. 1 Enabled.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the status register to generate an interrupt request.

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	0 Disabled. 1 Enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable Enables the TFIWF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the status register to generate a request. DSPIxRSER[RFDF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPIx_SR[RFDF] flag and DSPIx_RSER[RFDF_RE] must be set. 0 Interrupt request. 1 DMA request.
16 CMDFFF_DIRS	Command FIFO FIll DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPIx_SR[CMDFFF] flag and DSPIx_RSER[CMDFFF_RE] must be set. 0 Interrupt request. 1 DMA request.
17 DDIF_DIRS	DSI data received with active bits - DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPIx_SR[DDIF] flag and DSPIx_RSER[DDIF_RE] must be set. 0 Interrupt request. 1 DMA request.
18–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.3.7 DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)

PUSHR provides the means to write to the TX FIFO and CMD FIFO.

Data written to this register is transferred to:

- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.
- The CMD FIFO for writes to the Command field of PUSHR.

In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO.

In slave mode, CMD FIFO is unused and the 16-bit Command Field of PUSHR is reserved.

When Extended SPI Mode is not enabled (MCR[XSPI] = 0):

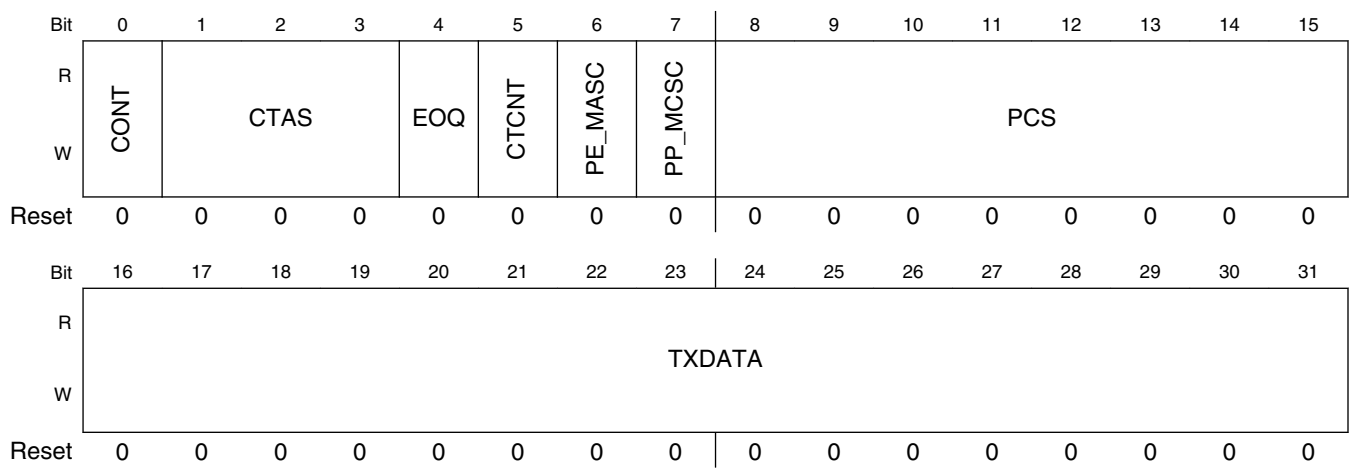
- TX FIFO and CMD FIFO must be filled simultaneously.
- Writes must be given to both Data and Command fields of PUSHR for every PUSHR operation.
- TX FIFO and CMD FIFO can be considered a single 32-bit FIFO.

When Extended SPI Mode is enabled (MCR[XSPI] = 1):

- TX FIFO and CMD FIFO can be written independently.
- A PUSHR Read Operation returns the topmost TX FIFO and CMD FIFO entries concatenated.

When DSPI Module is disabled, any writes to this register do not update the FIFO. Reads during Module disable mode return the last PUSHR write performed when Module was enabled.

Address: 0h base + 34h offset = 34h



DSPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable. (SPI master mode) Continuous Selection Format enables the selected PCS signals to remain asserted between transfers. PUSHR[EOQ] and PUSHR[CONT] bits cannot be asserted together in any Command Frame as EOQ signifies the last frame in a queue and the Last frame in a Continuous Transfer must have PUSHR[CONT] as 0. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select. Selects which CTAR (and corresponding CTARE register) to use in master mode to specify the transfer attributes for the associated SPI frame.

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

Field	Description
	In SPI slave mode, CTAR0 is used. See the Device Configuration chapter to determine how many CTARs this device has. Do not program a value in this field for a register that is not present. 000 CTAR0/CTARE0 001 CTAR1/CTARE1 010 CTAR2/CTARE2 011 CTAR3/CTARE3 100 CTAR4/CTARE4 101 CTAR5/CTARE5 110 CTAR6/CTARE6 111 CTAR7/CTARE7
4 EOQ	End Of Queue Host software uses this bit to signal to the DSPI that the current SPI transfer is the last in a queue. DSPIx_SR[EOQF] is set at the end of the transfer, . 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter. Clears the SPI_TCNT field in the Transfer Count register. The SPI_TCNT field is cleared before the DSPI starts transmitting the current SPI frame. Hence the current SPI frame causes this counter to increment by 1. 0 Do not clear the TCR[SPI_TCNT] field. 1 Clear the TCR[SPI_TCNT] field.
6 PE_MASC	Parity Enable or Mask t_{ASC} delay in the current frame PE: This bit enables parity bit transmission and parity reception check for the SPI frame. MASC: The current frame has the "after SCK" delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. This bit is used as Mask t_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE: No parity bit included/checked. MASC: t_{ASC} delay is not masked and the current frame has the after SCK delay. 1 PE: Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC: t_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask t_{CSC} delay in the next frame PP: controls the polarity of the parity bit transmitted and checked. MCSC: The next frame has the “PCS to SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. This bit is used as Mask t_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PP: Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame.

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

Field	Description
	MCSC: t_{CSC} delay is not masked and the next frame has the PCS to SCK delay. 1 PP: Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC: t_{CSC} delay is masked in the next frame.
8–15 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration chapter for the number of PCS signals used in this MCU. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command

49.3.8 DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO.

Data written to this register is transferred to:

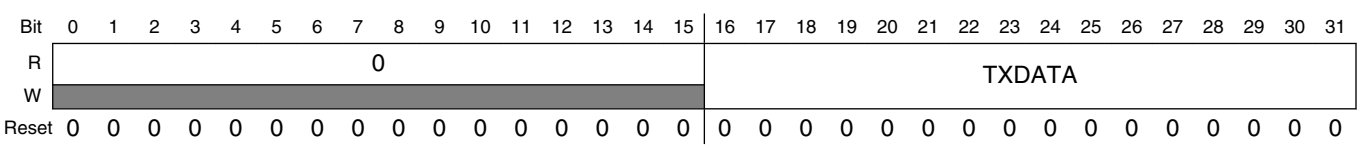
- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.

In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO.

In slave mode, CMD FIFO is unused and the 16-bit Command field of PUSHR is reserved.

In Slave mode, up to 32-bit SPI frames may be queued for transmission/reception by enabling MCR[XSPI] mode.

Address: 0h base + 34h offset = 34h



DSPI_PUSHR_SLAVE field descriptions

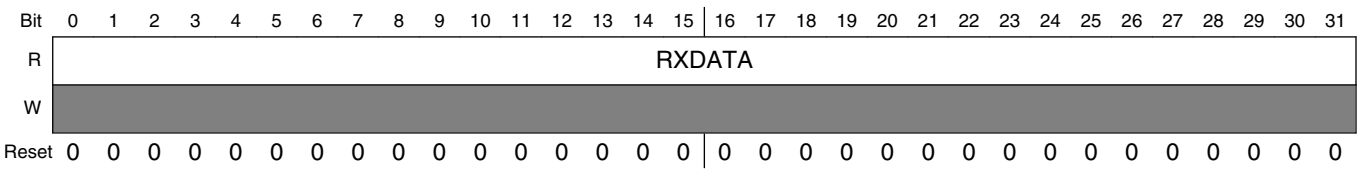
Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred.

49.3.9 DSPI POP FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO.

8- or 16-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register generates a Transfer Error.

Address: 0h base + 38h offset = 38h



DSPI_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

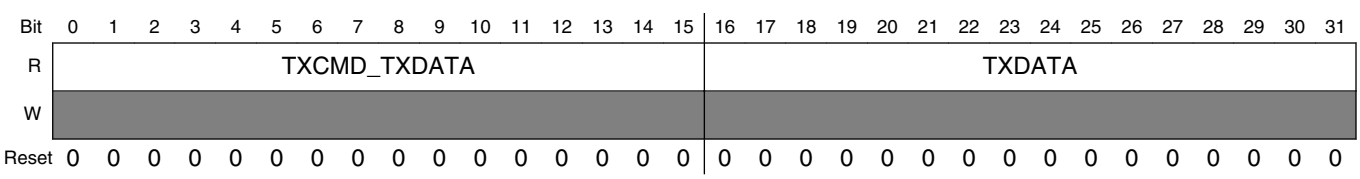
49.3.10 DSPI Transmit FIFO Registers (DSPI_TXFR*n*)

TXFRn provide visibility into the TX FIFO for debugging purposes.

Each read-only register is an entry in the TX FIFO. Reading the TXFRn registers does not alter the state of the TX FIFO.

Reading DSPIx_TXFRn registers is invalid for a DSPI Master when used in Extended SPI (XSPI) mode.

Address: 0h base + 3Ch offset + (4d × i), where i=0d to 3d



DSPI_TXFR*n* field descriptions

Field	Description
0–15 TXCMD_ TXDATA	Transmit Command or Transmit Data In master mode, the TXCMD field sets the transfer attributes for the SPI data. In slave mode, the TXDATA contains 16 MSB bits of the SPI data to be shifted out.
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

49.3.11 DSPI Receive FIFO Registers (DSPI_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes.

Each read-only register is an entry in the RX FIFO. Reading the RXFRn registers does not alter the state of the RX FIFO.

MCR[MDIS] must be 0 when RXFR is read.

Address: 0h base + 7Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_RXFR*n* field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

49.3.12 DSPI DSI Configuration Register 0 (DSPI_DSICR0)

DSICR0 selects various attributes associated with DSI and CSI Configurations.

Do not write to the DSICR0 while the DSPI is in the Running state.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	FMSZ4	Reserved						0	Reserved	ITSB	TSBC	TXSS	TPOL	Reserved	CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS			DMS	PES	PE	PP	DPCSx							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DSICR0 field descriptions

Field	Description
0 Reserved	This field is reserved.
1 FMSZ4	MSB of the frame size in master mode when DSI is used in 32-bit mode. If the bit is set, 16 is added to the frame size, as defined by the CTARn[FMSZ] field. The CTARx register is selected by the DSICTAS field.
2–7 Reserved	This field is reserved.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved.
10 ITSB	Interleaved TSB mode. Enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are sent when either the previous transmission was an SPI frame or the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. ITSB mode requires DSICR0[TSBC] set and should be written only when MCR[HALT] is asserted. If ITSB bit is set without setting DSICR0[TSBC], DSPI operates in Normal Mode depending on MCR[DCONF] bit. See Interleaved TSB (ITSB) Mode for more details. 0 Disabled. 1 Enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus Configuration, which allows up to 64-bit data to be used. It also allows t_{DT} to be programmable. 0 Disabled. 1 Enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either: a) data from host software written to the DSPI DSI Alternate Serialization Data Register (ASDR1/0), or b) parallel input pin states latched into the DSPI DSI Serialization Data Register (SDR1/0). 0 SDR source. 1 ASDR source.
13 TPOL	Trigger Polarity Selects the active edge of the hardware trigger input signal (HT), initiating DSI frames transfer. 0 Falling edge initiates a transfer. 1 Rising edge initiates a transfer.
14 Reserved	This field is reserved.
15 CID	Change In Data Transfer Enable

Table continues on the next page...

DSPI_DSICR0 field descriptions (continued)

Field	Description
	When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. Do not enable CID when DCONT is enabled. 0 Disabled. 1 Enabled.
16 DCONT	DSI Continuous Peripheral Chip Select Enable Enables the PCS signals to remain asserted between transfers. When the TSBC bit is set, DCONT has no effect. Do not enable DCONT when CID is enabled. 0 PCS signals return to inactive. 1 PCS signals remain asserted.
17–19 DSICTAS	DSI Clock and Transfer Attributes Select (DSI Master mode only) Selects the CTAR which provides transfer attributes for DSI frames. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop Stops DSI frame transmissions if DDIF flag is set in the status register. 0 Disabled. 1 Enabled.
21 PES	Parity Error Stop Stops DSI operation if a parity error occurred in the received DSI frame. 0 Disabled. 1 Enabled.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of 1 bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24–31 DPCSx	DSI Peripheral Chip Select 0-7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

49.3.13 DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Read-only SDR0 contains the states of the 32 LSB parallel input signals.

The states of these signals are latched into the SDR0 on the rising edge of every protocol clock.

When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SDR0 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB parallel input signals.

49.3.14 DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized.

When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames.

Writes to the ASDR0 take effect on the next frame boundary.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_ASDR0 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of data to be serialized.

49.3.15 DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Read-only COMPR0 holds a copy of the 32 LSB of last transmitted DSI data.

DSI data is transferred to this register as it is loaded into the TX Shift Register.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_COMPR0 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 LSB of last serialized DSI data.

49.3.16 DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Read-only DDR0 holds the 32 LSB signal states for the Parallel Output signals.

Host software can read data from incoming DSI frames.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DDR0 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data presented as signal states to the Parallel Output signals.

49.3.17 DSPI DSI Configuration Register 1 (DSPI_DSICR1)

DSICR1 selects various attributes associated with TSB (and ITSB) Configuration.

Do not write to the DSICR1 while the DSPI is in the Running state.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			TSBCNT					0					0	DSE1	DSE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRGPRD								DPCS1_x							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DSICR1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 TSBCNT	Timed Serial Bus Operation Count TSBCNT defines the length of the data frame When TSBC is set. The length value can vary from 3 to 31 (00011b to 11111b). The length value specifies the number of data bits to be shifted out during a transfer in TSB mode. The number of data bits in the data frame is one more than the value in the TSBCNT field.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 DSE1	Data Select Enable 1 When DSICR0[TBSC] bit is set, the DSE1 bit enables insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTARn register, selected by the DSICR0[DSICTAS] field. The TSBCNT field value must be greater than the FMSZ field value plus one for proper operation of the DSE1 bit. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When DSICR0[TSBC] bit is set, the DSE0 bit controls insertion of the zero bit (Data Select) in the beginning of the Data frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16–23 TRGPRD	Internal Trigger Period for the ITSB mode. When DSICR0[TRG] bit is cleared and ITSB Mode is enabled, this field determines the trigger period for the internal trigger in number of baud rate clock cycles. The trigger period should be calculated as:

Table continues on the next page...

DSPI_DSICR1 field descriptions (continued)

Field	Description
	Trigger Period = 32 bits (DSI frame maximum size) + 2 selection bits (maximum) + t_{PF} (passive frame time) Refer CTAR[FMSZ] field for minimum frame size allowed. See Interleaved TSB (ITSB) Mode for more details. 0 - 4 These values are invalid since minimum Command frame size is 5. (i.e. 1 selection bit + minimum frame size which is 4) 5 - 255 Number of baud rate clock cycles to be used as the trigger period. The trigger period implemented is equal to TRGPRD + 1.
24–31 DPCS1_x	DSI Peripheral Chip Select 0-7 These bits define the PCSs to assert for the second part of the DSI frame when operating in TSB (and ITSB) configuration with dual receiver. The DPCS1 bits select which of the PCS signals to assert during the second part of the DSI frame. The DPCS1 bits control the assertions of the PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

49.3.18 DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and from SDR0.

Each bit in the SSR0 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR0 register value has no effect.

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SSR0 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 LSB of DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0 The bit in the transmitted frame is taken from the parallel input pin. 1 The bit in the transmitted frame is taken from the ASDR0 register

49.3.19 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

DIMR0 selects bits in the 32 LSB of received DSI frame to be checked to generate the DDI interrupt.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DIMR0 field descriptions

Field	Description
0–31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame produces a DDI interrupt if the data bit matches the configured polarity.

49.3.20 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

DPIR0 defines which data bit value in the 32 LSB of received DSI frame generates the DDI interrupt.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DPIR0 field descriptions

Field	Description
0–31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

49.3.21 DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

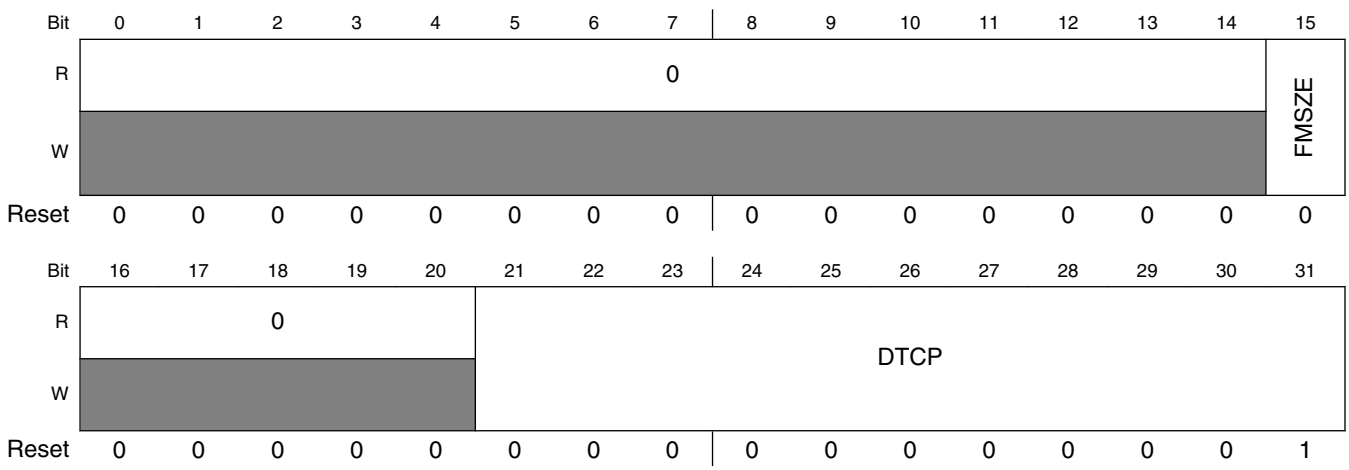
CTARE registers are used to define the extended transfer attributes for an SPI frame.

These registers are valid only when DSPIx_MCR[XSPI] is set.

When the DSPI is configured as:

- an SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used.
- an SPI bus slave, the CTARE0 register is used.

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 7d



DSPI_CTARE*n* field descriptions

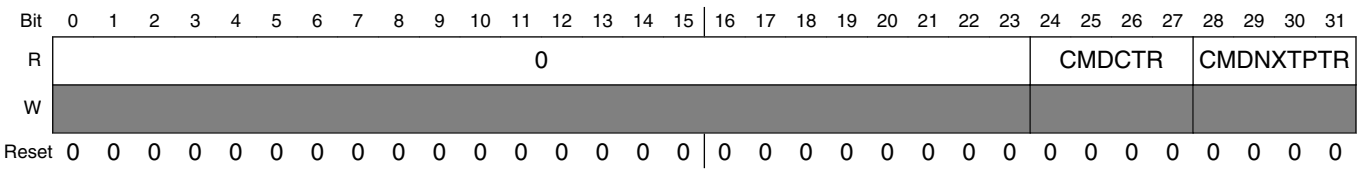
Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FMSZE	Frame Size Extended This field concatenated with DSPIx_CTAR[FMSZ] defines the frame size of the SPI frames to be transmitted. Frame size is the concatenation of {DSPIx_CTAR[FMSZ], DSPIx_CTARE[FMSZE]} + 1. 0 Default Mode. Up to 16-bit SPI frames can be transferred. 1 Up to 32-bit SPI frames can be transferred. Each frame transfer is a result of 2 TX FIFO Pops.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 DTCP	Data Transfer Count Preload This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame that selected this DSPIx_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

49.3.22 DSPI Status Register Extended (DSPI_SREX)

The DSPIx_SREX contains status fields that reflect the DSPI status and indicate the occurrence of events.

This register is not writable.

Address: 0h base + 13Ch offset = 13Ch



DSPI_SREX field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28–31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

49.4 Register classification for safety

The registers of this module are classified as per safety requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU. The registers containing these fields must be protected by periodic CRC or equivalent check implemented at chip level. The following register bits/fields are classified in this category:

- DSPI Module Configuration Register (DSPIx_MCR)
- DSPI Transfer Count Register (DSPIx_TCR)

- DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPIx_CTARn)
- DSPI Clock and Transfer Attributes Extended Register (In Master Mode) (DSPIx_CTAREn)
- DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)
- DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR) [0:15]
- DSPI DSI Configuration Register 0 (DSPIx_DSICR0)
- DSPI DSI Configuration Register 1 (DSPIx_DSICR1)

Safety Latent: The register bits/fields in this category should be protected by LBIST at chip level. The following register bits/fields are classified in this category:

- DSPI Status Register (DSPIx_SR) [0:15] (for stuck at 0)
- DSPI Status Register Extended (DSPIx_SREX) [0:8] (for stuck at 0)
- DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)
- DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)
- DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)
- DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)

Non-Safety: All other registers and bits fall in this category.

49.5 Functional description

The Deserial Serial Peripheral Interface (DSPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 64 parallel input/output signals. All communications are done with SPI-like protocol.

The DSPI has the following configurations:

- SPI Configuration in which the DSPI operates as a basic SPI or a queued SPI.
- DSI Configuration in which the DSPI serializes and deserializes Parallel Input/ Output signals or bits from memory-mapped register.
- CSI Configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the DSPI Configuration. See DSPI Module Configuration Register (DSPI_MCR) for the DSPI configuration values.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows CTAR selection on a frame by frame basis by setting a field in the SPI command. Extended SPI Mode (DSPIx_MCR[XSPI]) further allows the usage of CTAREn (CTARn Extended) registers to send multiple Data frames (of up to 32-bit frame size) using a single Command frame.

DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates which CTAR is used for the data transfer.

See DSPI Clock and Transfer Attributes Registers for information on the fields of the CTARs.

See DSPI Clock and Transfer Attributes Registers Extended for information on the fields of the CTARE registers.

Typical master to slave connections are shown in the following figure.

When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave: the data that was in the master shift register is now in the shift register of the slave, and vice versa.

At the end of a transfer, the TCF bit in the Status Register is set to indicate a completed transfer.

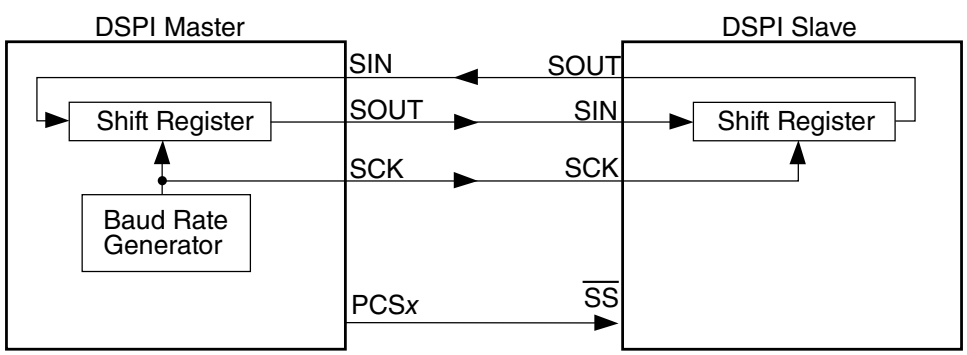


Figure 49-3. SPI serial protocol overview

Generally, more than one slave device can be connected to the DSPI master.

Eight Peripheral Chip Select (PCS) signals of the DSPI masters can be used to select which of the slaves to communicate with. Refer to the chip configuration chapter for the number of PCS signals used in this MCU.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in Transfer formats.

The transfer rate and delay settings are described in DSPI baud rate and clock delay generation.

49.5.1 Start and Stop of DSPI transfers

The DSPI has two operating states that are independent of DSPI configuration:

- STOPPED is the default DSPI state: no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results.
- RUNNING is the state when serial transfers take place.

DSPIx_SR[TXRXS] indicates what state the DSPI in (‘1’ = RUNNING state).

The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The DSPI stops (transitions from RUNNING to STOPPED) after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

49.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The SPI frames can be 32 bits long.

The DSPI is in SPI Configuration when the DCONF field in the MCR is 0b00.

The host CPU or a DMA controller transfers the SPI data from memory external to DSPI RAM queues to a transmit FIFO (TX FIFO) buffer.

Received data is stored in entries in the Receive FIFO (RX FIFO) buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the DSPI.

The FIFO buffers operation is described in Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) buffering mechanism and Receive First In First Out (RX FIFO) Buffering Mechanism. The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI Configuration supports two block-specific modes:

- Master mode: the DSPI initiates and controls the transfer according to the fields of the executing CMD FIFO entry.
- Slave mode: the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field space is reserved.

49.5.2.1 Master mode

In SPI Master mode, the DSPI initiates the serial transfers by controlling the Serial Communications Clock (SCK) and the Peripheral Chip Select (PCS) signals.

The executing SPI CMD FIFO entry determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See DSPI PUSH FIFO Register (PUSHR) for details on the SPI command fields.

The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin.

In SPI Master mode, each SPI frame to be transmitted has a command associated with it for transfer attribute control on a frame-by-frame basis.

In Extended SPI Master mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. Extended SPI Mode allows for larger frame sizes of up to 32 bits.

49.5.2.2 Slave mode

In SPI slave mode, the DSPI responds to transfers initiated by a SPI bus master. The DSPI does not initiate transfers.

Certain transfer attributes such as clock polarity, clock phase and frame size must be set in CTAR0 and CTARE0 for successful communication with an SPI master. The data is shifted out with MSB first; shifting out of LSB is not supported in this mode.

49.5.2.3 FIFO Disable operation

The FIFO Disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO.

The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled.

The Transmit and Receive side of the FIFOs are disabled separately:

- MCR[DIS_TXF] disables TX FIFO and CMD FIFO.
  - TFFF, TFUF, CMDFFF, CMDCTR and TXCTR fields in the status register and the extended status register behave as if there is a one-entry FIFO.
  - The contents of the TXFR registers and TXNXTPTR and CMDNXTPTR are undefined.
- MCR[DIS_RXF] bit disables the RX FIFO.
  - The RFDF, RFOF and RXCTR fields in the status register behave as if there is a one-entry FIFO.
  - The contents of the RXFR registers and POPNXTPTR are undefined.

The FIFO Disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

49.5.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission.

The TX FIFO holds 4 words, each consisting of SPI data that is added to the TX FIFO by writing to the Data field of DSPI PUSH FIFO Register (PUSHR).

The number of entries in the TX FIFO is device-specific.

TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the TX FIFO. TXCTR is updated every time an 8 or 16-bit write takes place to the Data field of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. This field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

The TXFRn Registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

49.5.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set.

When TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete, the TFFF bit is cleared. Writing a ‘1’ to the TFFF bit also clears it. The DSPI ignores attempts to push data to a full TX FIFO; the state of the TX FIFO does not change and no error condition is indicated.

The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill interrupt or DMA request for details.

49.5.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled and the frame size of SPI Data to be transmitted is more than 16 bits, then two Data entries are popped from TX FIFO simultaneously and transferred to the shift register. The first of the two popped entries forms the 16 LSB bits of the SPI frame. Such an operation also causes TX FIFO Counter to decrement by two.

At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a ‘1’ to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a DSPI slave while the slave’s DSPI TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave’s SR is set. See Transmit FIFO Underflow interrupt request for details.

49.5.2.5 Command First In First Out (CMD FIFO) buffering mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission.

The CMD FIFO holds four entries, each representing command fields. The number of entries in the CMD FIFO is device-specific.

SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is disabled:

- The TX FIFO and CMD FIFO must be filled together as every CMDFIFO entry has a corresponding single TXFIFO entry attached to it.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled:

- The TX FIFO and CMD FIFO can be filled independently as every CMDFIFO entry can have multiple TXFIFO entries attached to it.
- The CTARE[DTCP] field decides the number of SPI Data Frames of size {FMSZE, FMSZ} to be transmitted using the current Command Entry. FMSZ and FMSZE fields are given in the CTAR/CTARE registers respectively, pointed by the CTAS field in the Command frame.
- The amount of time a command entry is in use is known as a Command Cycle. The Busy flag DSPIx_SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place on the lower half of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The CMDNXTPTR field indicates which CMD FIFO entry is used during the next command cycle. This field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

The TXFRn Registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

49.5.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin.

The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific.

SPI data is added to the RX FIFO on completion of a transfer when the received data in the shift register is transferred into the RX FIFO.

SPI data are removed (popped) from the RX FIFO by reading the DSPI POP FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the status register points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR register number and it rolls over after reaching the maximum.

49.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register.

While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the status register is set to indicating an overflow condition. Consequently:

- If the DSPIx_MCR[ROOE] bit is set, the incoming data is shifted in to the shift register.
- If the DSPIx_MCR[ROOE] bit is cleared, the incoming data is ignored.

49.5.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the DSPI POP FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one.

Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data read from the empty RX FIFO is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the status register is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete, or by writing a '1' to it.

49.5.3 Deserial Serial Interface (DSI) configuration

The DSI Configuration supports pin count reduction by serializing parallel input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in Transfer formats. The received serial frames are converted to a parallel form (deserialized) and placed on the Parallel Output signals or in the DSPI_DDR. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR1/0).

The DSI frames can be from 4 to 32 bits.

49.5.3.1 DSI master mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has four different conditions (described in DSI transfer initiation control) that can initiate a transfer:

- Continuous.
- Change in data.
- Trigger signal.
- Trigger signal combined with a change in data.

Transfer attributes are set during initialization. DSICR0[DSICTAS] field determines which of the DSPI_CTARs will control the transfer attributes.

49.5.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers.

Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1.

The data is shifted out with MSB first.

49.5.3.3 DSI serialization

In the DSI Configuration, up to 32 bits can be serialized using two different sources. The TXSS bit in the DSPI_DSICR0 selects between the DSPI DSI Serialization Data Register (DSPI_SDR0) and the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0) as the source of the serialized data. The DSPI_SDR0 holds the latest 32 parallel input signal values which is sampled at every rising edge of the Bus clock. The DSPI_ASDR0 register is written by host software and used as an alternate source of serialized data.

The DSPI has to sample the inputs (from SDR0 or ASDR0) before it transmits a DSI frame. Due to the asynchronous clocking structure within DSPI, the sampled data (from SDR0 or ASDR0) must be synchronized into the Protocol Domain before being loaded into the Shift Register for transmission.

To prevent Clock Domain Crossing issues, inputs from SDR0 or ASDR0 should not change while they are being sampled into the Protocol Domain. This is ensured in the following manner for the ASDR0 inputs. The initial write performed on ASDR0 register will cause the 32-bit ASDR0 input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain. Once synchronized, the lock is removed and the next ASDR0 write will again trigger this process. If there was a write performed on ASDR0 while the lock was asserted, this will be taken into consideration as soon as the previous synchronization process completes and lock gets de-asserted. In case,multiple writes were performed on ASDR0 when lock was asserted, only the latest value will be considered for synchronization once the lock gets de-asserted.

The entire process explained above is also true for the SDR0 input, the only difference being that SDR0 inputs are delayed internally by a single Bus clock (register interface clock) to detect a change in the parallel input pins.

Once the synchronization process for a particular De-serial input begins, it takes up to 5 Bus clock (register interface clock) periods and 5 Protocol clock periods before the synchronization process for the next De-serial input can begin. Hence the writes performed on ASDR0 and SDR0 are constrained with this timing.

Note

The Parallel Input pin state, to be transmitted, should be selected by TXSS or DSPI_SSR0 register and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR0 register provides additional way to create the frame for transmission. Each bit from this register is OR'd with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR0 and DSPI_ASDR0 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. Control field may come from DSPI_ASDR0 register, set by the device's CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift Register is stored in the DSPI DSI Transmit Comparison Register (DSPI_COMPR0). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. The following figure shows the DSI Serialization logic.

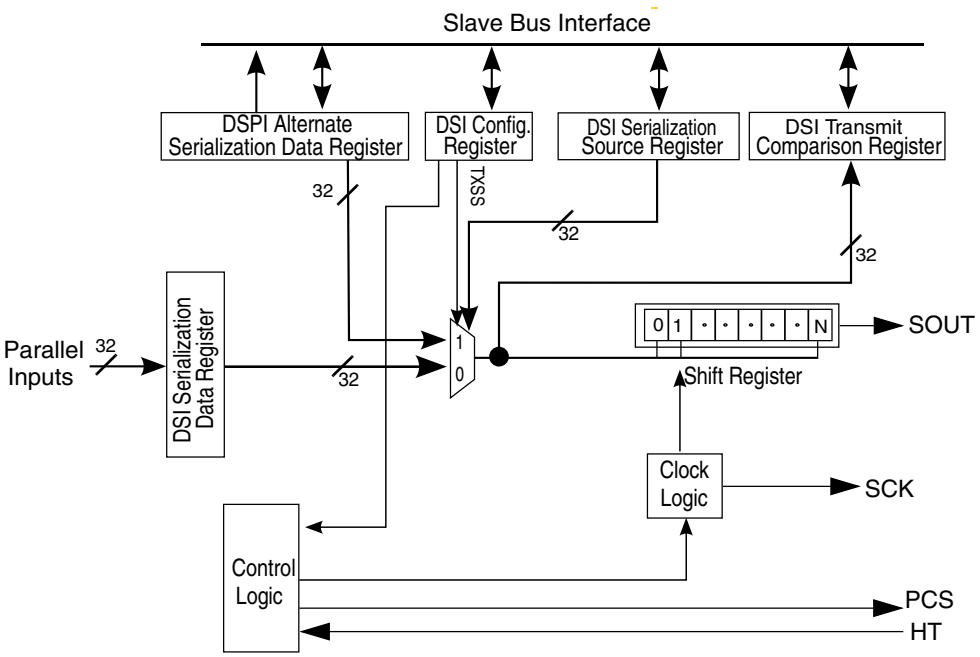


Figure 49-4. DSI serialization diagram

49.5.3.4 DSI deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR0-1). This register presents the deserialized data as Parallel Output signal values. The DSPI_DDR0-1is memory-mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR0-1), bit-wise ANDed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR0-1) and the results ORed to produce DDIF flag in the

DSPI_SR. Which in turn can cause DDIF interrupt request or DMA request based on RSER[DDIF_RE] and RSER[DDIF_DIRS] and/or stop DSI frame transmissions if DSICR0[DMS] bit is set. Figure 49-5 shows the DSI Deserialization logic.

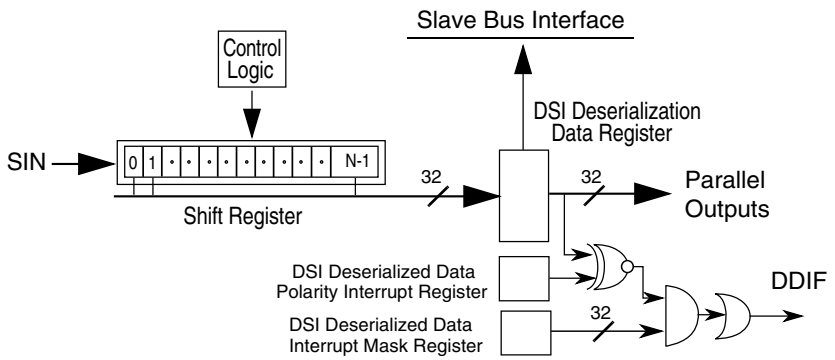


Figure 49-5. DSI deserialization diagram

49.5.3.5 DSI transfer initiation control

Data transfers for a master DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the CID bits in the DSPI_DSICR0 register. The following table lists the four transfer initiation conditions.

DSPI_DSICR0 bits	Transfer initiation control
CID	
0	Continuous
1	Change in Data
0	Triggered
1	Triggered or Change in Data

49.5.3.5.1 Continuous control

For continuous control, a new DSI frame shifts out when the previous transfer cycle has completed and the delay after transfer (t_{DT}) has elapsed.

49.5.3.5.2 Change in data control

For change in data control, a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in the DSPI_COMPR0. When the data selected for the transfer from the DSPI_SDR0 and DSPI_ASDR0 registers is different from the data in the DSPI_COMPR1/0 a new DSI frame is transmitted.

49.5.3.5.3 Triggered control

For Triggered Control, initiation of a transfer is controlled by the Hardware Trigger signal (HT). DSICR0[TPOL] bit selects the active edge of HT. For HT to have any affect, DSICR0[TRRE] bit must be set.

49.5.3.5.4 Change in data or triggered control

For change in data or triggered control, initiation of a transfer is controlled by the detection of a change in data to be serialized or by the HT signal.

It is not advisable to configure a DSPI Slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1; since the following statement then holds true:

Once slave select is asserted, there is a very small window (of up to three module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occurs, the next frame to be transmitted would be loaded into the shift register. Since it is not feasible that such a write is initiated and completed within this short span of time, the data transmitted by DSPI slave would lag by one data frame.

49.5.4 Combined Serial Interface (CSI) configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis.

CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the Parallel Output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. Figure 49-6 shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

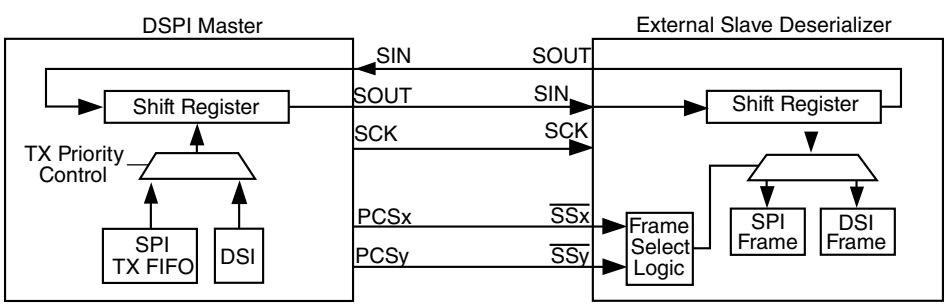


Figure 49-6. Example of System using DSPI in CSI Configuration

In CSI Configuration, the DSPI transfers DSI data based on DSI transfer initiation control. When there are SPI commands in the TX FIFO, the SPI data has priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTARs associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy.

The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the Parallel Output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

49.5.4.1 CSI serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR selected by the DSICR0[DSICTAS] field.

The parallel inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR0) on the rising edge of every protocol clock and serialized based on the transfer initiation control settings in the DSPI_DSICR0. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty), they have priority over DSI data and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI_COMPR0. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

49.5.4.2 CSI deserialization

The deserialized frames in CSI Configuration goes into the DSPI_DDR0 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred, the returned frames are deserialized and latched into the DSPI_DDR0. When SPI frames are transferred, the returned frames are deserialized and written to the RX FIFO.

49.5.5 DSPI baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

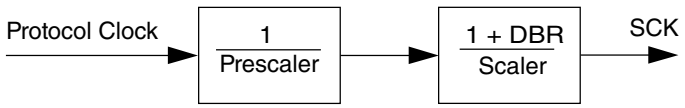


Figure 49-7. Communications clock prescalers and scalers

49.5.5.1 Baud rate generator

The baud Rate is the frequency of the Serial Communication Clock (SCK). The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 49-5. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

Note

The clock frequencies mentioned in the table above are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.5.5.2 PCS to SCK delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See the following table for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTARx registers select the PCS to SCK delay by the formula in the CSSCK field description. Table 49-6 shows an example of how to compute the PCS to SCK delay.

Table 49-6. PCS to SCK delay computation example

f_P	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCSCSK and CSSCK fields have no effect in TSB configuration.

Note

The clock frequency mentioned in the table above is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.5.5.3 After SCK delay (t_{ASC})

The After SCK delay is the length of time between the last edge of SCK and the negation of PCS. See the following table for an illustration of the After SCK delay. The PASC and ASC fields in the CTARx registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 49-7. After SCK delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCASC and ASC fields have no effect in TSB configuration.

Note

The clock frequency mentioned in the table above is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.5.5.4 Delay after transfer (t_{DT})

The delay after transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See the following table for an illustration of the delay after transfer. the PDT and DT fields in the CTARx registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the delay after transfer.

Table 49-8. Delay after transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

Note

The clock frequency mentioned in the table above is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in non-continuous clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] bitfield description. When in continuous clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode, the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See detailed information on Timed Serial Bus (TSB).

49.5.5.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The PCSS signal provides a delay to allow the PCS signals to settle after a transition occurs, thereby avoiding glitches. When the DSPI is in master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[0]–PCS[4]and PCS[6]–PCS[7] signals into as many as 128 glitch-free PCS signals. Figure 49-8 shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

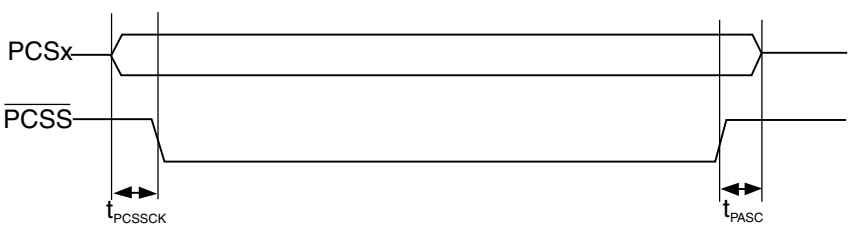


Figure 49-8. Peripheral Chip Select Strobe Timing

The delay between the assertion of the PCS signals and the assertion of PCSS is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_P} \times \text{PCSSCK}$$

Equation 22. Equation for t_{PCSSCK}

At the end of the transfer the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{PASC} = \frac{1}{f_P} \times PASC$$

Equation 23. Equation for t_{PASC}

Table 49-9 shows an example of how to compute the t_{PCSSCK} delay.

Table 49-9. Peripheral Chip Select Strobe Assert Computation Example

f_P	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

Table 49-10 shows an example of how to compute the t_{PASC} delay.

Table 49-10. Peripheral Chip Select Strobe Negate Computation Example

f_P	PASC	Prescaler	Delay after transfer
100 MHz	0b11	7	70.0 ns

The $\overline{PCSS}$ signal is not supported when Continuous Serial Communication SCK or TSB mode are enabled.

Note

The clock frequency mentioned in the preceding Table 49-9 and Table 49-10 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.5.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL—Selects the idle state polarity of the SCK.
- CPHA—Selects if the data on SOUT is valid before or on the first SCK edge.

Even though the bus slave does not control the SCK signal, in slave mode these values must be identical to the master device settings to ensure proper transmission. In SPI slave mode, only CTAR0 is used.

In DSI slave mode, only CTAR1 is used.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0.
- Classic SPI with CPHA = 1.
- Modified Transfer format with CPHA = 0.
- Modified Transfer format with CPHA = 1.

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI and DSI Configurations, the DSPI provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

49.5.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in the following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

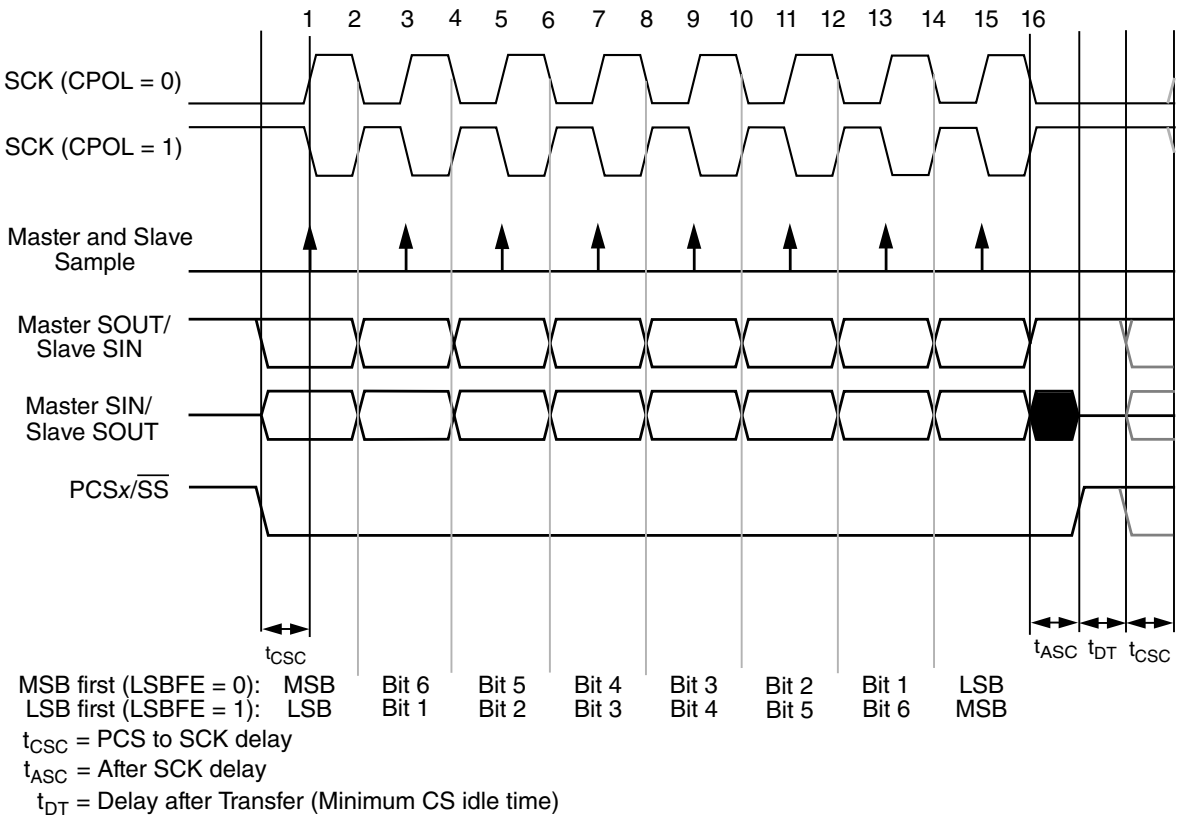


Figure 49-9. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin.

After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals.

At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals.

For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges.

After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

49.5.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges

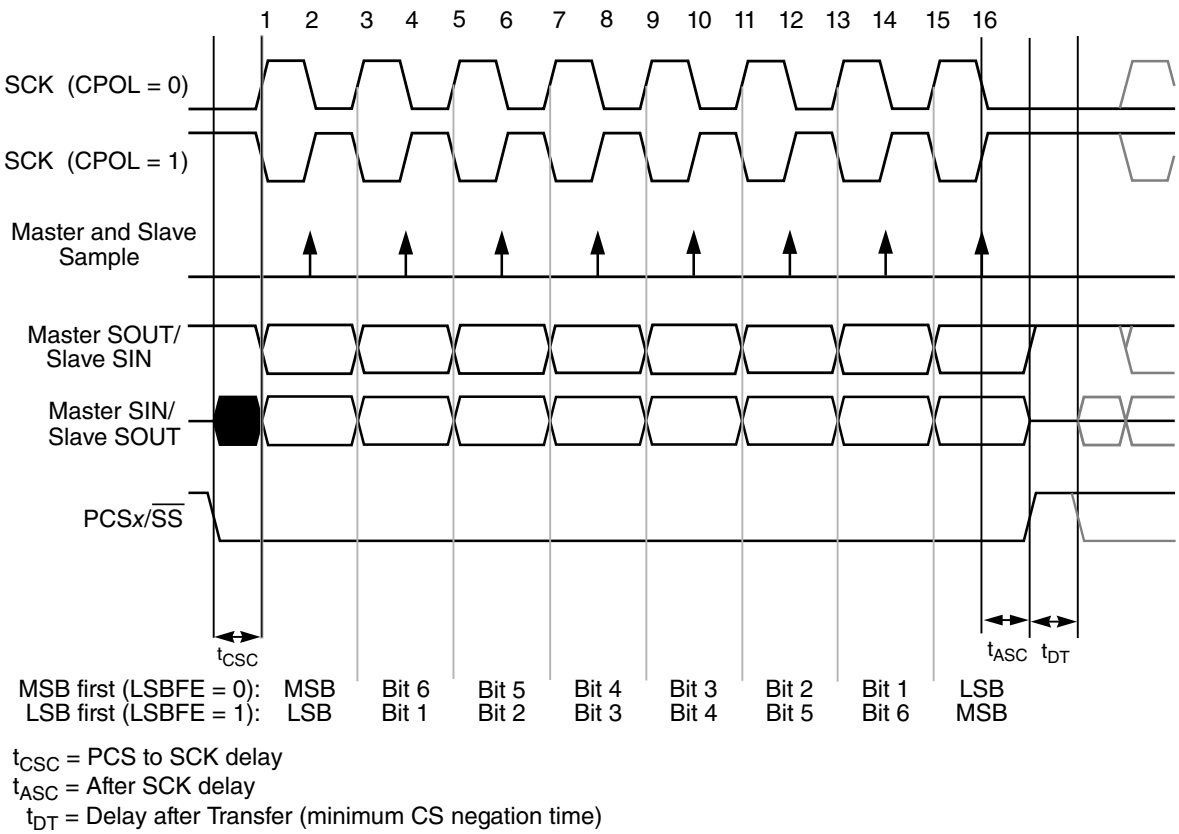


Figure 49-10. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)

The master initiates the transfer by asserting the PCS signal to the slave.

After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins.

For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges.

After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

49.5.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actually sample point is delayed by 1 protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE = 1. Timing delays shown are:

- t_{CSC} – PCS to SCK assertion delay
- t_{ACS} – After SCK PCS negation delay
- t_{su_ms} – master SIN setup time
- t_{hd_ms} – master SIN hold time
- t_{vd_sl} – slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- t_{su_sl} – data setup time on slave data input
- t_{hd_s}l – data hold time on slave data input
- t_p – protocol clock period

Figure 49-11 shows the modified transfer format for CPHA = 0 and f_p/f_{sck} = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behaviors are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_p represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

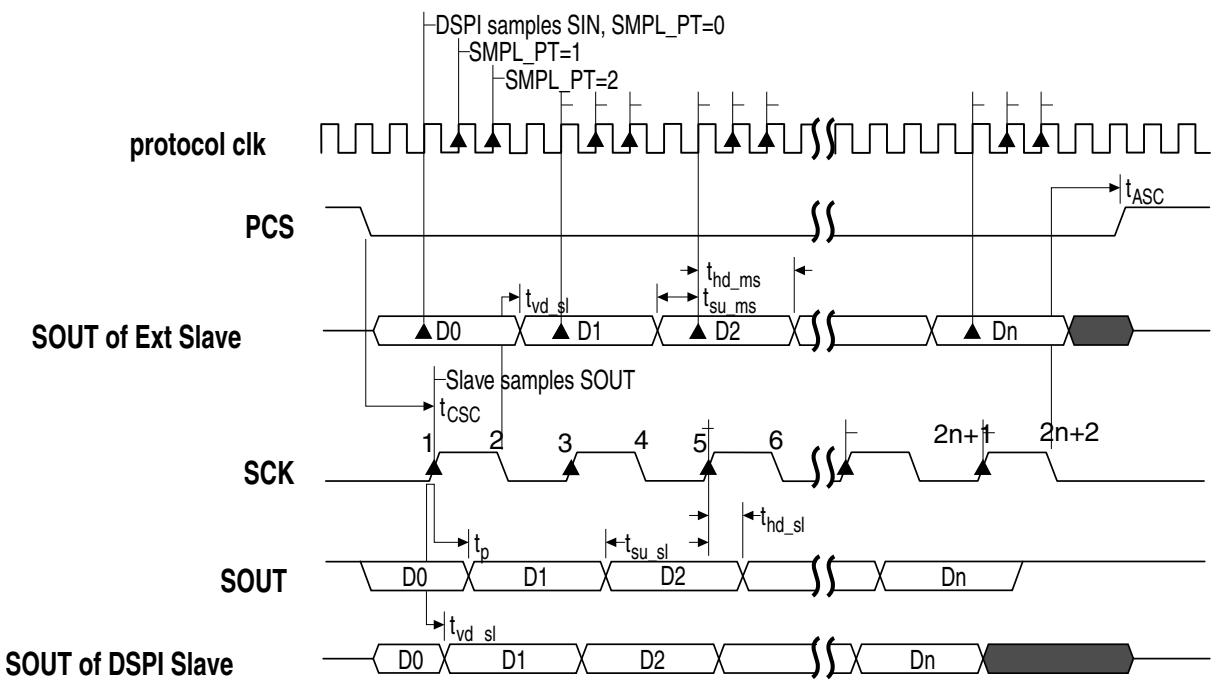


Figure 49-11. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)

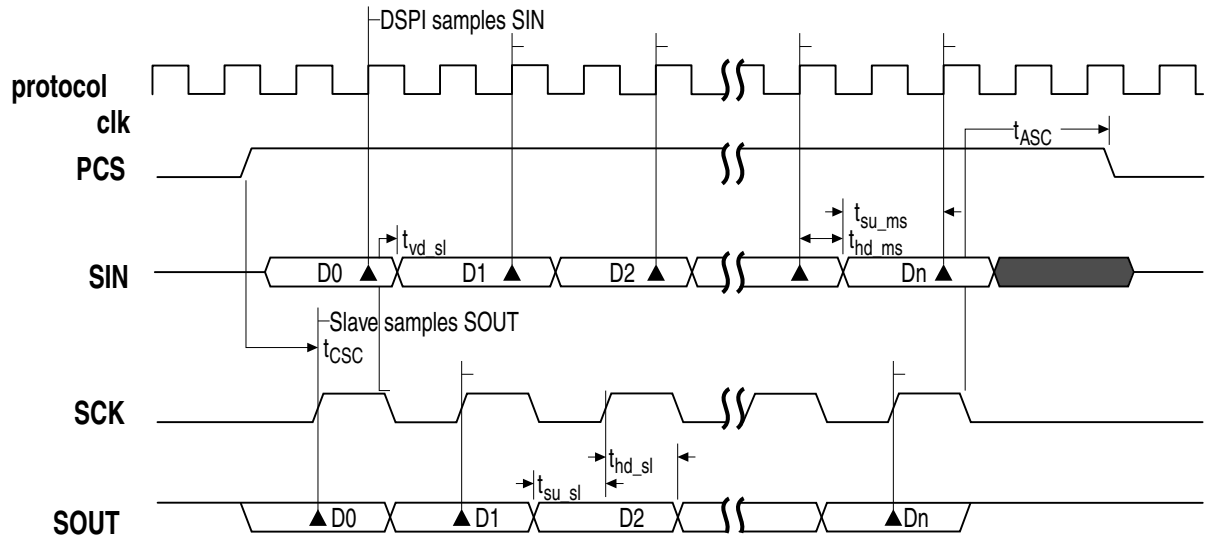


Figure 49-12. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

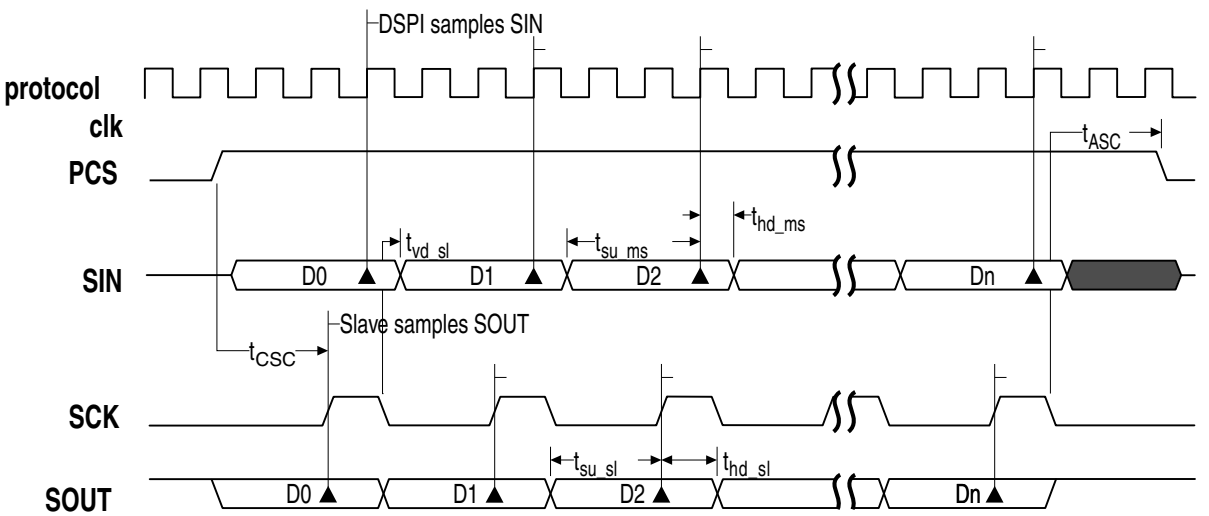


Figure 49-13. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)

49.5.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT

bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The After SCK (tASC) delay must be greater or equal to half of the SCK period.**

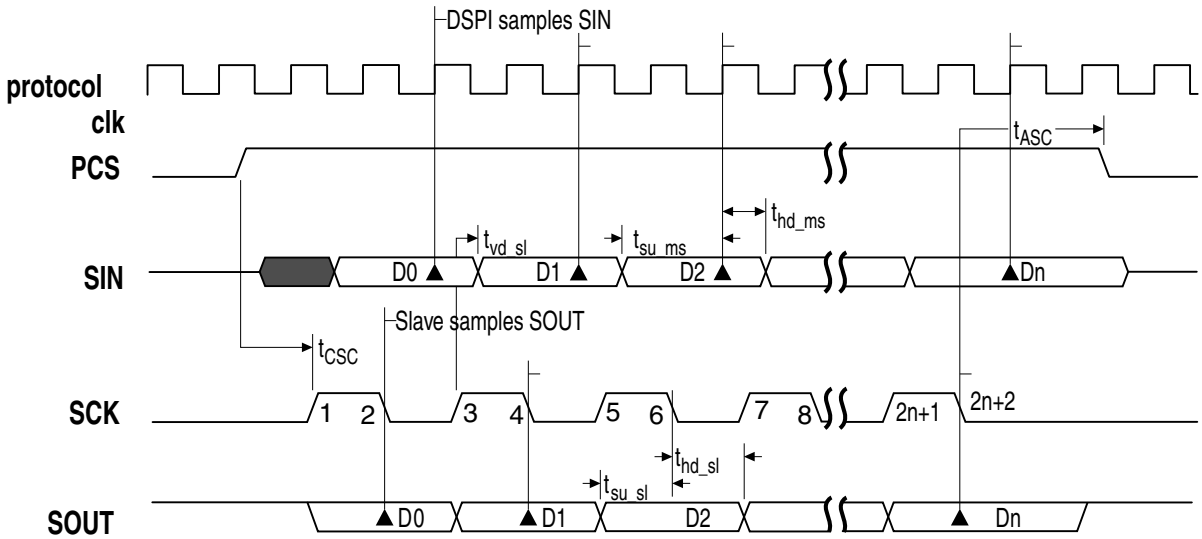


Figure 49-14. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)

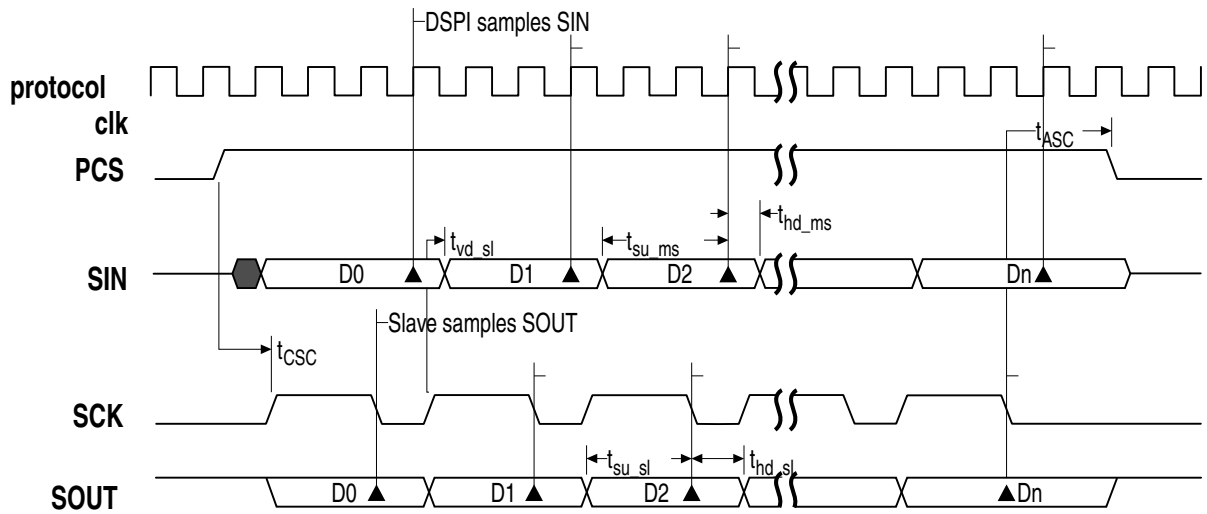


Figure 49-15. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)

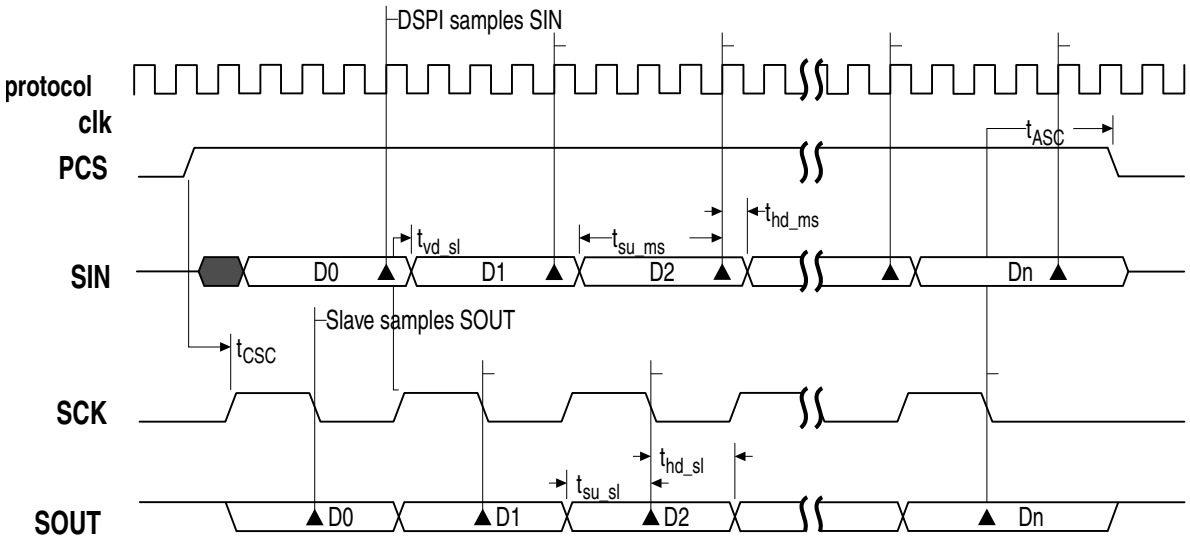


Figure 49-16. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)

49.5.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI Configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0.

The behavior of the PCS signals in the two configurations is identical so only SPI Configuration will be described. When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The timing diagram in Figure 49-17 is for two four-bit transfers with CPHA = 1 and CONT = 0.

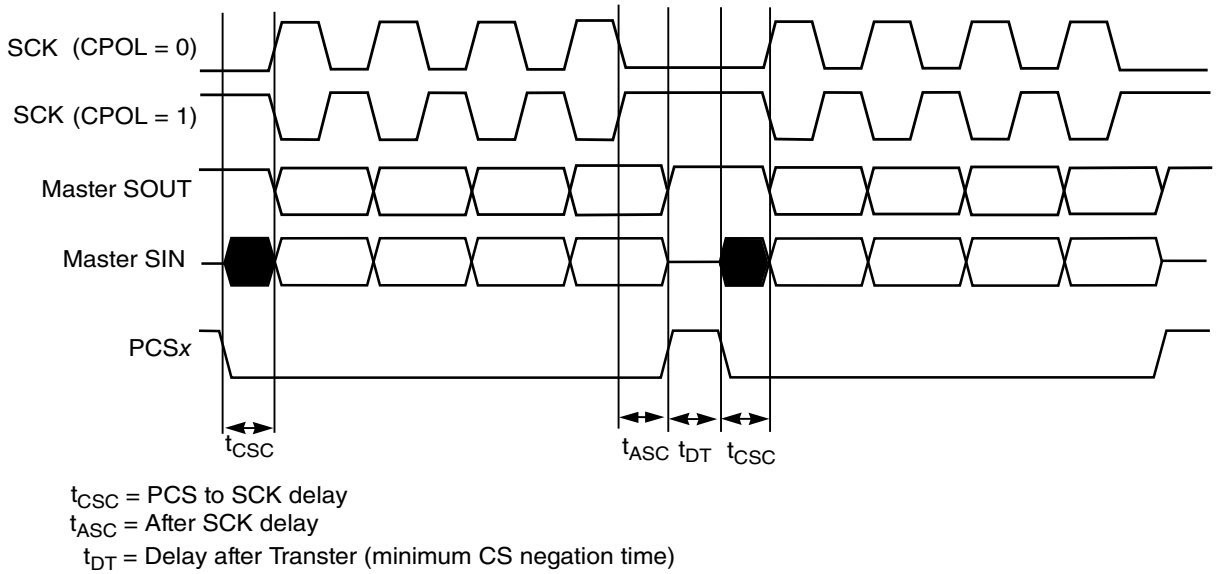


Figure 49-17. Example of non-continuous format (CPHA = 1, CONT = 0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. Figure 49-18 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

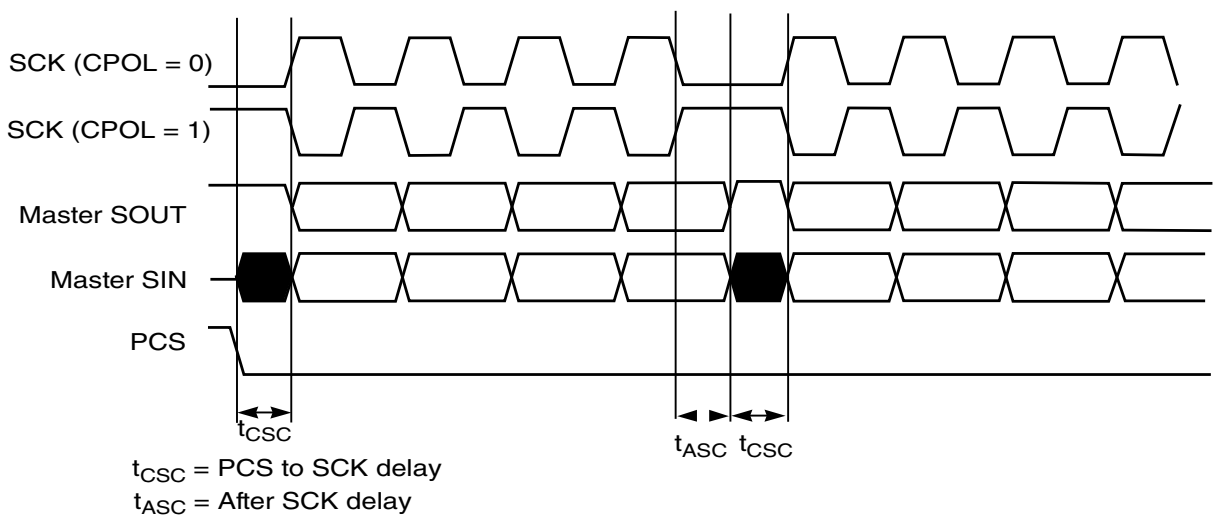


Figure 49-18. Example of continuous transfer (CPHA = 1, CONT = 1)

When using DSPI with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs selected by transmit commands must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit de-asserted (in master mode) and the user software must provide sufficient frames in the TX_FIFO to be sent out (in slave mode) and the master de-asserts the PCSn at end of transmission of last frame.
- The PUSHR[CONT]/DSICR0[DCONT] bits must be de-asserted before asserting MCR[HALT] bit (in master mode). This ensures that the PCSn signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from RUNNING to STOPPED state.

Note

You must fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty.

When operating in slave mode, ensure that when the last-entry in the TXFIFO is completely transmitted (that is the corresponding TCF flag is asserted and TXFIFO is empty), the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

49.5.6.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to the Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

Note

The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

The Fast Continuous Selection Format is enabled by writing '1' into FCPCS bit of the MCR. When this bit is asserted, MASC and MCSC bits of the PUSHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the "After SCK" delay for the current frame.
- MCSC bit masks the "PCS to SCK" delay for the next frame.
- "After SCK" (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The "PCS to SCK" delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here, too.

The following figure shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

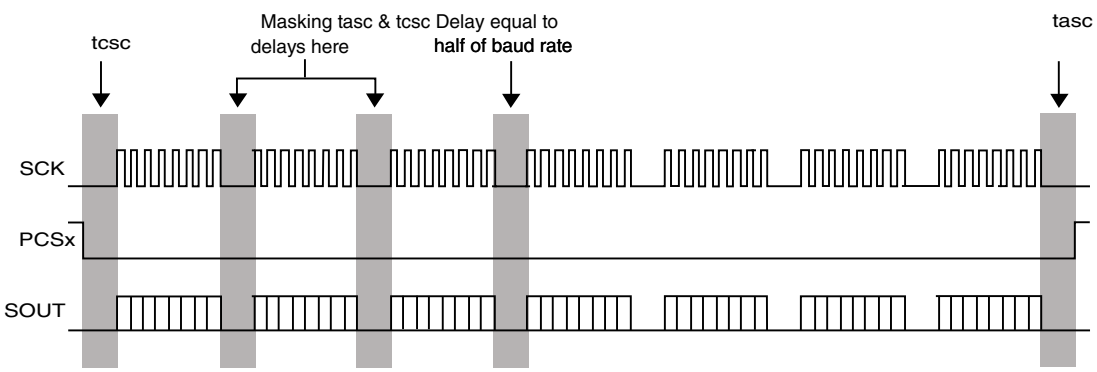


Figure 49-19. Example of Fast Continuous Selection Format

In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

49.5.7 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the DSPI is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the DSPI is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.
    - At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
    - At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle.

When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles. Figure 49-20 is the timing diagram for Continuous SCK format with Continuous Selection disabled.

Note

When in Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TXFIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

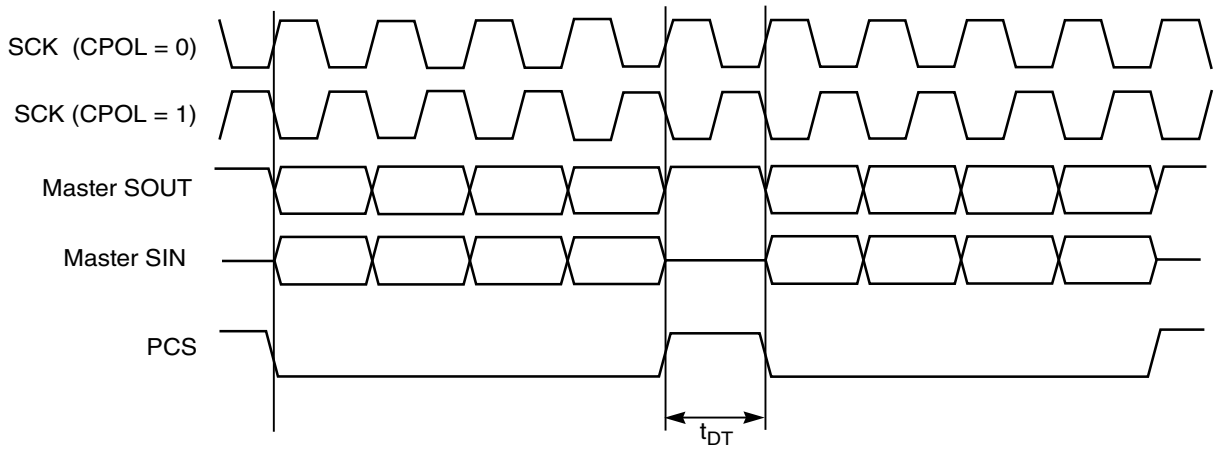


Figure 49-20. Continuous SCK timing diagram (CONT = 0)

If the CONT bit in the TX FIFO entry is set or the DCONT in the DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (refer to Start and Stop of DSPI transfers.
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 49-21 shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

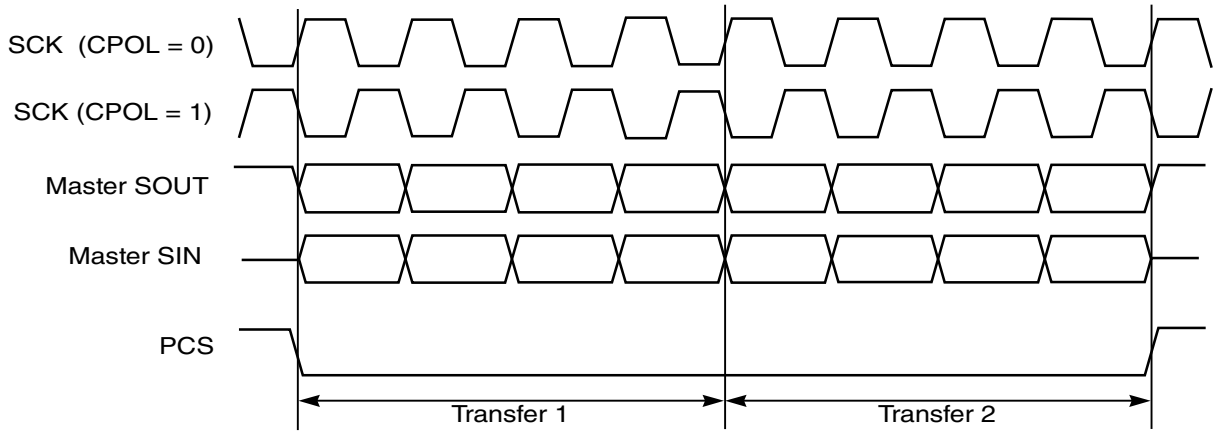


Figure 49-21. Continuous SCK Timing Diagram (CONT = 1)

49.5.8 Slave mode operation constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the DSPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and SS signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR1/0 register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

49.5.9 Timed Serial Bus (TSB)

The DSPI can be programmed in Timed Serial Bus configuration by setting DSICR0[TSBC] bit. See DSPI DSI Configuration Register 0 (DSPI_DSICR0) for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the device.

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

Figure 49-22 shows the signals used in the TSB interface.

In the TSB configuration the DSPI is able to send from 4 to 34 bits MSC data frames (4 to 32 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0), written by the host software,
- parallel input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR0).

DSPI_DSICR0 TXSS bit or DSPI_SSR0 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

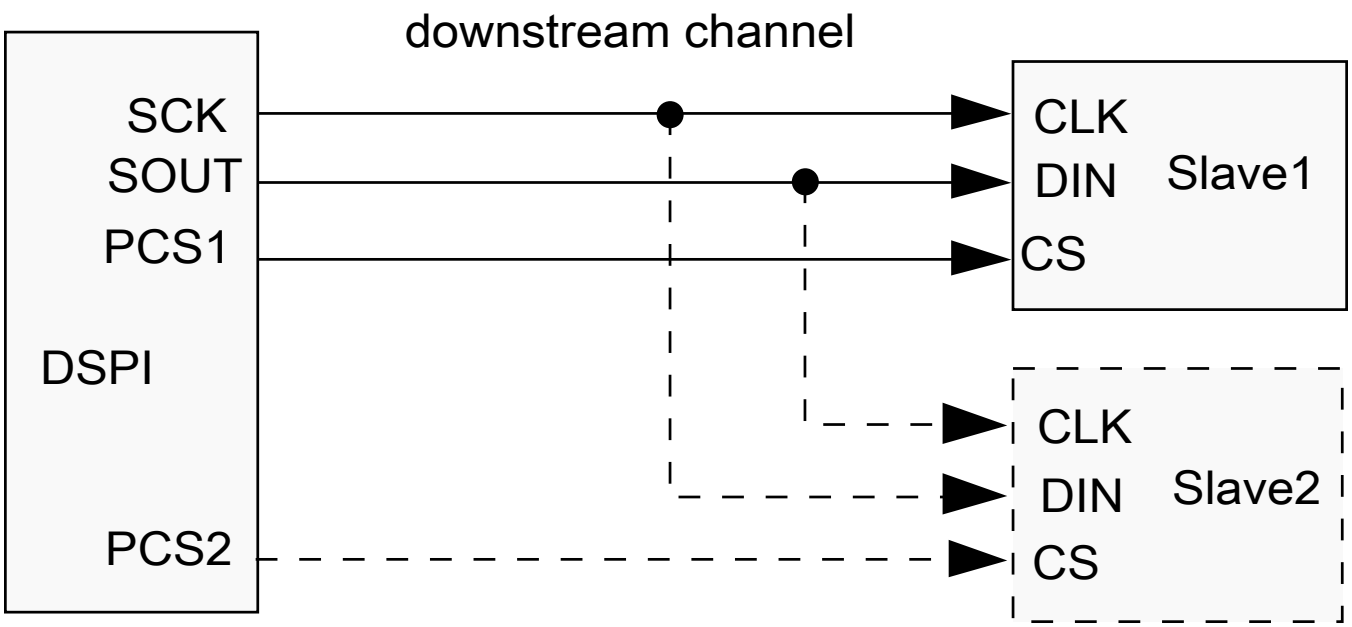


Figure 49-22. DSPI usage in the TSB Configuration

The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTARn PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

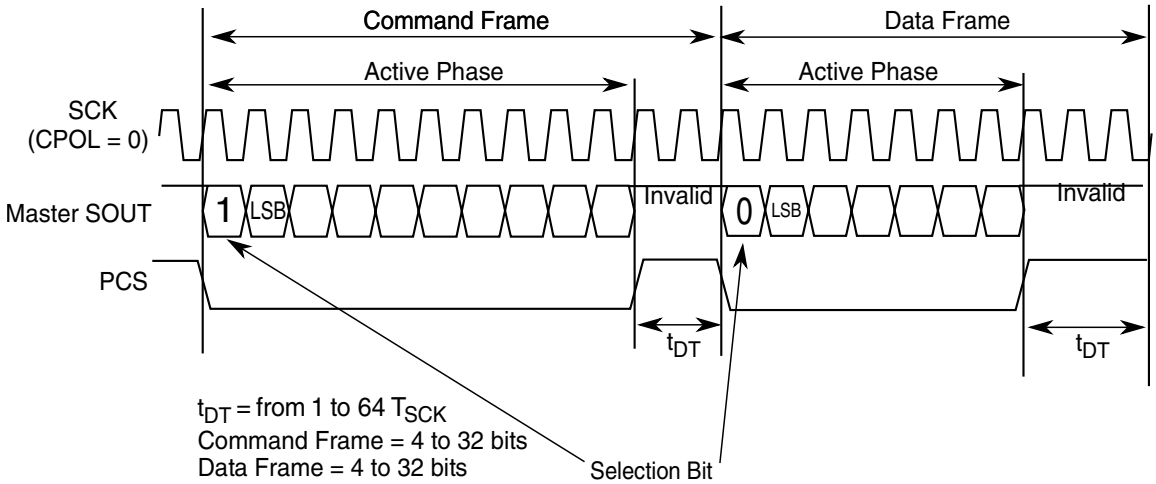


Figure 49-23. TSB Downstream Frames

The above figure shows the two types of MSC downstream frames: command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit "0" indicates a data frame
- The selection bit "1" indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI].

The data frame consists of up to 32 bits from the SDR0 or ASDR0 registers and up to two selection bits (0). Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSEn bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTARn[LSBFE] to transmit least significant bit first.

Regardless of the LSBFE bit setting, the Data Frame Selection Bits, if enabled, are always transmitted first, before the corresponding data subframes.

49.5.9.1 MSC Dual Receiver Support with PCS Switchover

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switchover occurs, is defined by FMSZ field of the DSPI_CTARn register, selected by DSICTAS field of the DSICR0 register.

NOTE

When using Dual receiver support, it must be ensured that the number of bits to be transmitted to each slave must be at least 4.

Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCSn bits, after that by DSPI_DSICR1 DPCS1_n bits.

The PCS switchover occurs at the driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switchover if enabled.

The Data Frame with PCS switchover is shown in the following figure.

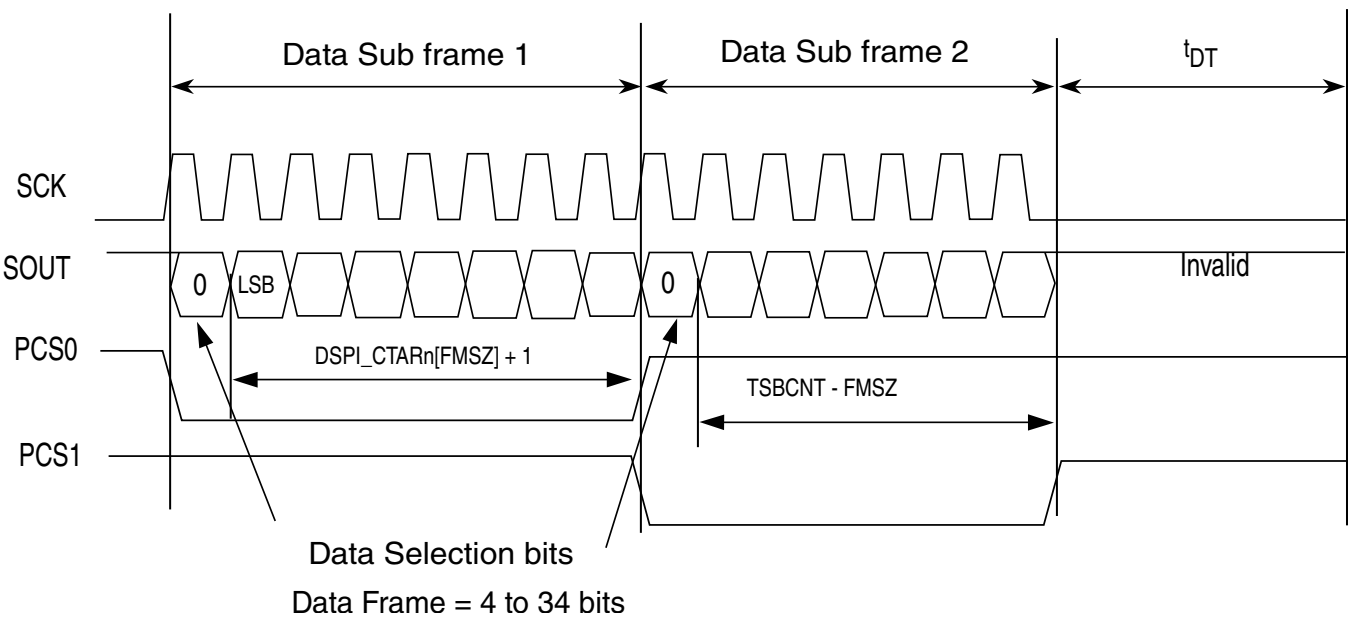


Figure 49-24. TSB Data Frame Format for MSC Dual Receiver Operation

49.5.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB in all aspects except for the differences mentioned in Table 49-11. In the Interleaved TSB or ITSB mode, frame transmission is governed by the following rules:

- Transmission of frames is executed on a periodic trigger that is generated internally to the DSPI. The trigger period is calculated as follows:
  - Trigger Period = 32 bits (maximum data frame size) + 2 selection bits (maximum bits in data frame) + t_{PT}
    - t_{PT} is the passive frame time or dead time inserted to have constant trigger period.
  - The user software shall configure the trigger period and trigger source.
- If the previously transmitted frame was a SPI frame or the TX FIFO (or CMD FIFO) is empty, the frame transmitted on subsequent trigger will always be a DSI frame.
- If the previously transmitted frame was not a SPI frame and the TX_FIFO (and CMD FIFO) is not empty, the frame transmitted on subsequent trigger will always be a SPI frame.

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) in such a way to prevent back to back SPI frames. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits cannot be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

Table 49-11 lists the differences between the TSB and ITSB modes.

Table 49-11. Differences between TSB and ITSB modes

Mode attribute	TSB mode	ITSB mode
Frame priority	Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Frame selection bit	Selection bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
Frame transmission	Occasionally triggered transmission (that is trigger is received via hardware trigger input).	Periodically triggered transmission (that is frames are transmitted continuously on every trigger).
Frame gap	Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is inserted automatically as the whole operation is based on a periodic trigger. If a frame has less than 32-bits to be transmitted, dead or passive time is automatically inserted by waiting for the next trigger pulse. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

In TSB and ITSB mode, separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI. MSC dual receiver support with PCS switchover is also supported in ITSB mode. See MSC Dual Receiver Support with PCS Switchover for more details.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the ITSB mode features listed in the table above.

49.5.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.
- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. DSICR0[CID] bit should be cleared. DSICR0[DCONT] should not be used in ITSB mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame completion (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in Interleaved TSB (ITSB) Mode.

49.5.10.2 Using ITSB mode for MSC downstream channel

MSC Downstream Channel uses two types of frames: command frames and data frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- If command frame has less bits than data frame, a gap is inserted to ensure next data frame starts at known reference time.

- In dual MSC ASIC, some bits of the data frame (which can be up to 64 bits) goes to first ASIC and remainder bits to second ASIC.
- In dual MSC ASIC, if a command frame is sent to one ASIC, it must not change the timing of the next data frame.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (See Configuring DSPI for ITSB mode), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in Combined Serial Interface (CSI) configuration.

The following figure shows a sample MSC Downstream transmission using the ITSB mode.

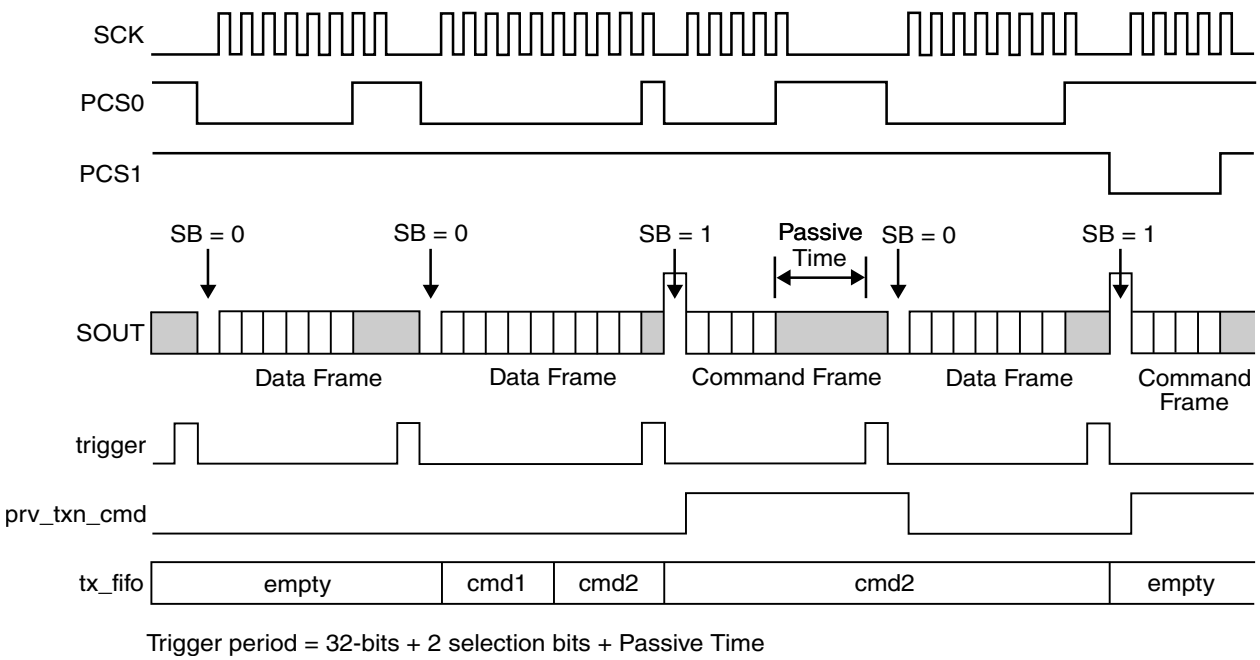


Figure 49-25. Sample MSC downstream transmission using ITSB mode

Brief description of operation includes:

- At the first trigger, since there are no command frames, a data frame is transmitted: to send a command frame at the start of operation, the user software must program a command frame in the TX_FIFO before enabling DSPI (MCR[HALT] = 0).
- The second trigger sets off the transmission of another data frame.
- Between second and third trigger, two command frames are pushed into the TX_FIFO.

- At third trigger, a command frame is transmitted as the TX_FIFO is not empty: since only 16 bits or less are transmitted, no transmission happens till next trigger. This is marked as the "Passive Time". The passive time is added in all frames after frame transmission completes and until next trigger pulse is detected.
- At fourth trigger, since the previous transmission was a command frame, a data frame is transmitted.
- At fifth trigger, since one command frame is pending in TX_FIFO and last transmitted frame was not a command frame, a command frame is transmitted.
- At sixth trigger (not shown), the process continues.

49.5.11 Parity generation and check

The DSPI module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The register fields setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The DSPI module can be programmed to stop SPI or/and DSI frame transmission in case of a frame reception with parity error.

49.5.11.1 Parity for SPI frames

When the DSPI is in the master mode the parity generation is controlled by PE and PP bits of the TX FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the DSPI stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode, the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

49.5.11.2 Parity for DSI frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. (DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode, the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation, clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

49.5.12 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. The following table lists these conditions.

Table 49-12. Interrupt and DMA request conditions

DSPI condition	Flag	Request type	
		Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	—
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	—
Transfer Complete	TCF	Yes	—
CMD Transfer Complete	CMDTCF	Yes	—
SPI Transfer Complete	SPITCF	Yes	—
DSI Transfer Complete	DSITCF	Yes	—

Table continues on the next page...

Table 49-12. Interrupt and DMA request conditions (continued)

DSPI condition	Flag	Request type	
		Interrupt	DMA
TX FIFO Underflow	TFUF	Yes	—
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	—
SPI Parity Error	SPEF	Yes	—
DSI Parity Error	DPEF	Yes	—
DSI Deserialized Data Match	DDIF	Yes	Yes

Each condition has a flag bit in the DSPI Status Register (SR) and a Request Enable bit in the DSPI DMA/Interrupt Request Select and Enable Register (RSER). The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the RSER.

The DSPI module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

49.5.12.1 End of Queue interrupt request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and the EOQF_RE bit in the RSER is set.

When MCR[XSPI] is enabled and the EOQ bit in the executing SPI command is set, the End of Queue Request will only be generated once the last Data frame in the Command Cycle has been transmitted.

Note

This interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

49.5.12.2 Transmit FIFO Fill interrupt or DMA request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

Note

TFFF flag clears automatically when DMA is used to fill TXFIFO.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TXFIFO:

1. Wait until TFFF = 1
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a '1' to its location. If TX FIFO is not full, this flag will not clear.

49.5.12.3 Command FIFO Fill interrupt or DMA request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled and hence TX FIFO and CMD FIFO can be filled independently. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

Note

CMDFFF flag clears automatically when DMA is used to fill CMD FIFO.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMD FIFO:

1. Wait until CMDFFF = 1
2. Write data to Command field of PUSHR using CPU.
3. Clear CMDFFF by writing a '1' to its location. If CMD FIFO is not full, this flag will not clear.

49.5.12.4 Transmit FIFO Invalid Write interrupt request

The Transfer Fifo Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transfer Fifo Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

49.5.12.5 Transfer Complete interrupt request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

49.5.12.6 Command Transfer Complete interrupt request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

49.5.12.7 SPI Transfer Complete interrupt request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

49.5.12.8 DSI Transfer Complete interrupt request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

49.5.12.9 Transmit FIFO Underflow interrupt request

The Transmit FIFO Underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the DSPI, operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

49.5.12.10 Receive FIFO Drain interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

49.5.12.11 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

49.5.12.12 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

49.5.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

49.5.12.14 Deserialized Data Match Interrupt or DMA Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR0 data, masked with DSPI_DIMR0, has been received. This Request is generated if DDIF_RE bit in the DSPI_RSER is set. The DDIF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

49.5.13 Power saving features

The DSPI supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

NOTE

When in Slave Mode, the module can only be put in Low Power Modes (such as STOP and HALT) once the Master module has transitioned to the Low Power Mode as well.

49.5.13.1 Stop mode (External Stop Mode)

The DSPI supports the stop mode protocol. When a request is made to enter External Stop mode, the DSPI block acknowledges the request. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

If a DSPI block is configured in Slave mode, External Stop mode requests must be run after the last clock cycle or Low-power mode is not entered.

49.5.13.2 Module disable mode

Module disable mode is a block-specific mode that the DSPI can enter to save power. The host CPU can initiate the module disable mode by setting the MDIS bit in the MCR. The module disable mode can also be initiated by hardware.

When the MDIS bit is set, the DSPI negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, DSPI is said to have entered Module Disable Mode. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.

If implemented, the Clock Enable signal can stop the clock to the non-memory-mapped logic. When Clock Enable is negated, the DSPI is in a dormant state, but the memory-mapped registers are still accessible.

Certain read or write operations have a different effect when the DSPI is in the module disable mode:

- Reading the RX FIFO Pop Register does not change the state of the RX FIFO.
- Writing to the FIFO Push Register does not change the state of the TX FIFO or CMD FIFO.
- Clearing of the FIFOs has no effect.
- Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect.
- All status bits and register flags in the DSPI return the correct values when read, but writing to them has no effect.
- Writing to the TCR during module disable mode has no effect.
- Interrupt and DMA request signals cannot be cleared.

49.6 Initialization/application information

This section describes how to initialize the DSPI module.

49.6.1 Managing DSPI queues

The queues are not part of the DSPI, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI Configuration.

1. When DSPI executes last command word from a queue, the EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is cleared to indicate the STOPPED state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.

5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a '1' to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

49.6.2 Switching master and slave mode

When changing modes in the DSPI, follow the steps below to guarantee proper operation.

1. Halt the DSPI by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a '1' to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable the DSPI by clearing MCR[HALT].

49.6.3 Initializing DSPI in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the DSPI is enabled by clearing MCR[HALT]. It should be ensured that DSPI Slave is enabled before enabling DSPI Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

49.6.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is clear.

Note

The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the clocking chapter for the frequency used to drive this module in the device.

Table 49-13. Baud rate values (bps)

		Baud rate divider prescaler values (DSPI*n*_CTAR[PBR])			
		2	**3**	**5**	**7**
Baud Rate scaler values (DSPI*n*_CTAR[BR])	**2**	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
	4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
	6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
	8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
	16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
	32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
	64	781 kHz	521 kHz	312 kHz	223 kHz
	128	391 kHz	260 kHz	156 kHz	112 kHz
	256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
	512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
	1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
	2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
	4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
	8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
	16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz
	32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz

49.6.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

Note

The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

Table 49-14. Delay values

		Delay prescaler values (DSPI*n*_CTAR[PBR])			
		1	**3**	**5**	**7**
Delay scaler values	**2**	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

49.6.6 Calculation of FIFO pointer addresses

Complete visibility of the TX and CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped.

- For TX FIFO, the first-in pointer is the Transmit Next Pointer (TXNXTPTR).
- For CMD FIFO, the first-in pointer is the Command Next Pointer (CMDNXTPTR).
- For RX FIFO, the first-in pointer is the Pop Next Pointer (POPNXTPTR).

Figure 49-26 illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts are applicable to the CMD FIFO and RX FIFO.

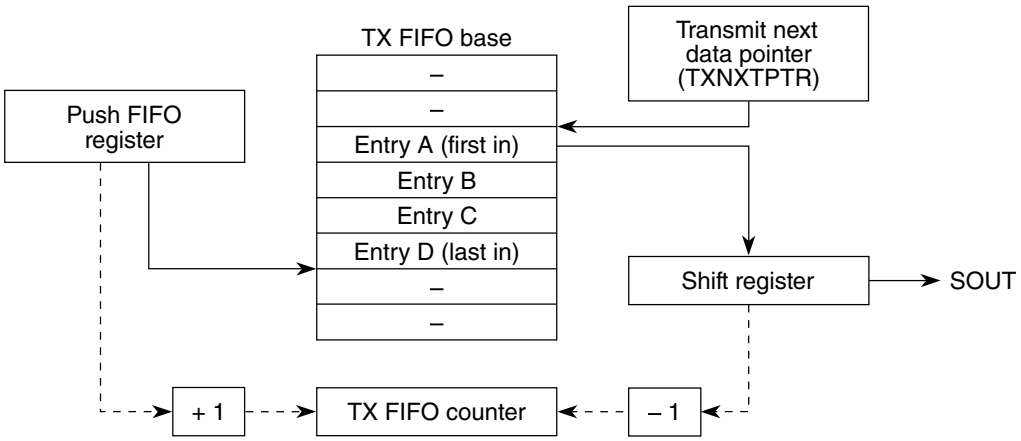


Figure 49-26. TX FIFO pointers and counter

See Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) buffering mechanism and Receive First In First Out (RX FIFO) Buffering Mechanism for details on the FIFO operation.

49.6.6.1 Address calculation for the first-in entry and last-in entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in Entry Address} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

Equation 24. Equation for first-in entry address

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-in Entry Address} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFO depth})$$

Equation 25. Equation for last-in entry address

TX FIFO Base – Base address of TX FIFO

TXCTR – TX FIFO Counter

TXNXTPTR – Transmit Next Pointer

TX FIFO Depth – Transmit FIFO depth, implementation specific

49.6.6.2 Address calculation for the first-in entry and last-in entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in Entry Address} = \text{CMD FIFO Base} + (4 \times \text{CMD NXTPTR})$$

Equation 26. Equation for first-in entry address

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-in Entry Address} = \text{CMD FIFO Base} + 4 \times (\text{CMD CTR} + \text{CMD NXTPTR} - 1) \bmod (\text{CMD FIFO depth})$$

Equation 27. Equation for last-in entry address

CMD FIFO Base – Base address of CMD FIFO

CMDCTR – CMD FIFO Counter

CMDNXTPTR – Command Next Pointer

CMD FIFO Depth – Command FIFO depth, implementation specific

49.6.6.3 Address calculation for the first-in entry and last-in entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in Entry Address} = \text{RX FIFO Base} + (4 \times \text{POPNXTPTR})$$

Equation 28. Equation for first-in entry address

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-in Entry Address} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFO depth})$$

Equation 29. Equation for last-in entry address

RX FIFO Base – Base address of RX FIFO

RXCTR – RX FIFO counter

POPNXTPTR – Pop Next Pointer RX FIFO Depth – Receive FIFO depth, implementation specific

Chapter 50
Zipwire

50.1 Overview

The SIPI and LFAST modules work together as a single unit called Zipwire. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules.

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

50.2 Introduction

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals and/or memories of another MCU or smart ASIC. Zipwire is implemented in hardware so there is no CPU load for the intitiator or target node. Zipwire supports 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address at the target node.

Zipwire architecture is fully pipelined and support multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock for a total of five pins.

Zipwire also supports a streaming mode for the transfer of large blocks of data.

In streaming mode Zipwire has a high transfer rate. For single random access, 32-bit read, from any 32-bit address location, the latency is approximately 1 µs.

50.3 Zipwire Block Diagram

Figure 50-1 shows two MCUs with Zipwire connected using a five wire interface. Both MCUs support Target and Initiator modes of Zipwire. The SIPI Bus Master Interface is used for all Target mode transactions. The SIPI Register Interface is used for all Initiator Transactions and initial setup. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST reference clock coming from MCU B clock system. The Ref_Clock can come from either MCU. It is user configurable, but it must be the same clock that drives both LFAST modules.

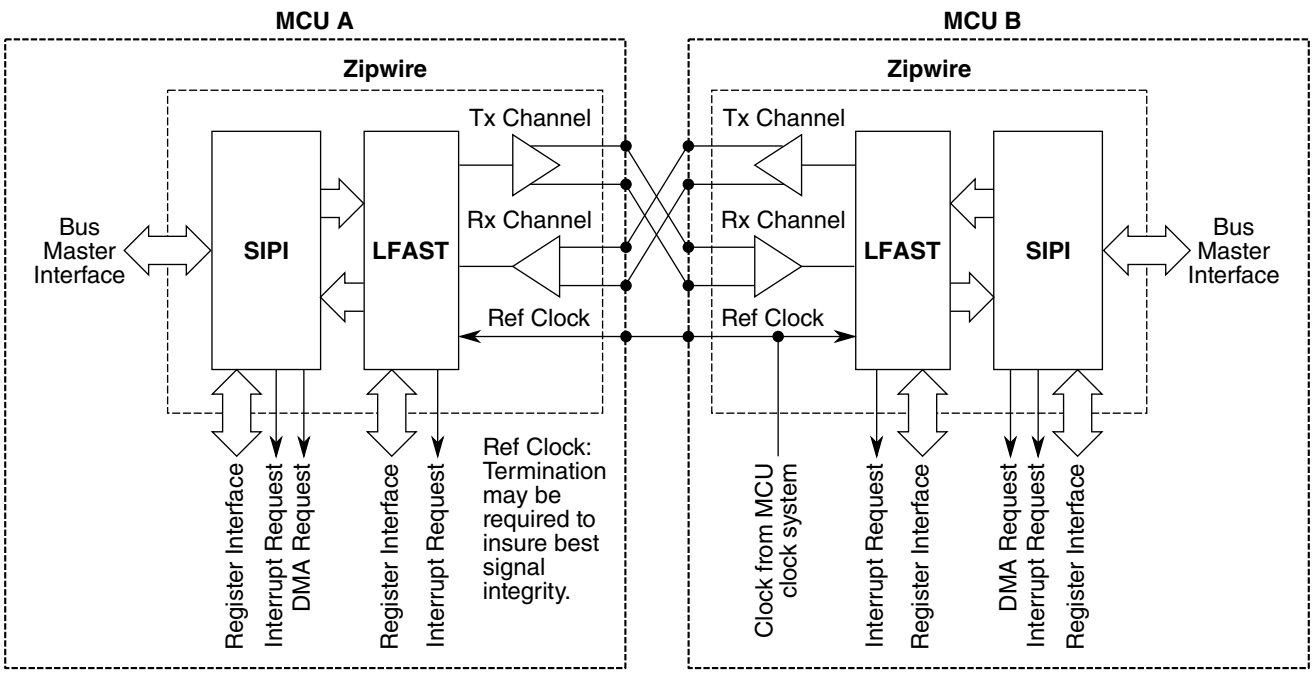


Figure 50-1. Zipwire connection diagram

50.4 Architecture

Zipwire is composed of several modules and system resources. The physical layer is LVDS with 1.2 V common mode voltage and 200mV swing.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.

- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). SIPI runs on top of LFAST and is fully encapsulated within the LFAST payload. SIPI uses four fixed sizes of LFAST payload in Zipwire:

- 32-bit
- 64-bit
- 96-bit
- 288-bit

The SIPI protocol implements a suite of commands initiated by one MCU, for reading and writing any 32-bit address location in another connected MCU. The SIPI protocol allows either or both MCUs to initiate commands. The protocol also supports interrupt requests from one MCU to the other.

The SIPI module implements both the initiator part of the protocol and the recipient part of the protocol. The module implements four SIPI Initiator channels that can run independently of each other and can run simultaneously. The SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the inititating MCU, SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

The initiator part of the module is connected to the DMA controller and is able to generate DMA requests as a command is completed. This allows a series of read or write commands to be queued in the Inititator MCU and executed by DMA and SIPI without CPU intervention at either Initiator or Target MCU.

50.5 Zipwire interconnections

LFAST has the following connections:

- Tx and Rx configuration is controlled by LFAST Control registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.

- Tx Data Port/Rx Data Port — Directly connected to the SIPI module. Allows received data to be efficiently transferred to SIPI and transmit data to be transferred from SIPI.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers.
- DMA connections — Allows SIPI command sequences to be queues and initiated without CPU intervention.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

50.6 Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Inititator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

50.6.1 Read performance

A Zipwire read operation consists of three stages:

- Inititator sends SIPI Read command to Target.

- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data
- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in a LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes 278 ns (= 89 × 3.13 ns).

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of 6 × slow XBAR cycles to read different memories.
- Slow XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, time for SIPI to read an address location is:

(13 × 50 MHz clocks) + (6 × 100 MHz clocks) = 320 ns

50.6.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

Read Bandwidth = 4 bytes in 320 ns = 12.5 MB/second

50.6.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

Read Latency = 278 ns + 320 ns + 278 ns = 876 ns

50.6.2 Write performance

A Zipwire write operation consists of three stages:

- Initiator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address

- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header
- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32-bit (acknowledge)
- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes 378 ns
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes 178 ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of 5 × slow XBAR cycles to write different memories.
- Slow XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, the time for SIPI to write an address location is:

(13 × 50 MHz clocks) + (5 × 100 MHz clocks) = 310 ns

50.6.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = 4 bytes in 378 ns = 10.6 MB/second

50.6.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = 378 ns + 310 ns + 178 ns = 866 ns

Chapter 51
Serial Interprocessor Interface (SIPI)

51.1 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol which runs on top of the LFAST (LVDS Fast Asynchronous Serial Transmission) module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

51.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between 2 devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:
  - Full Implementation = 4
  - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum implementation either Initiator or Target.
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface

- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS)

51.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

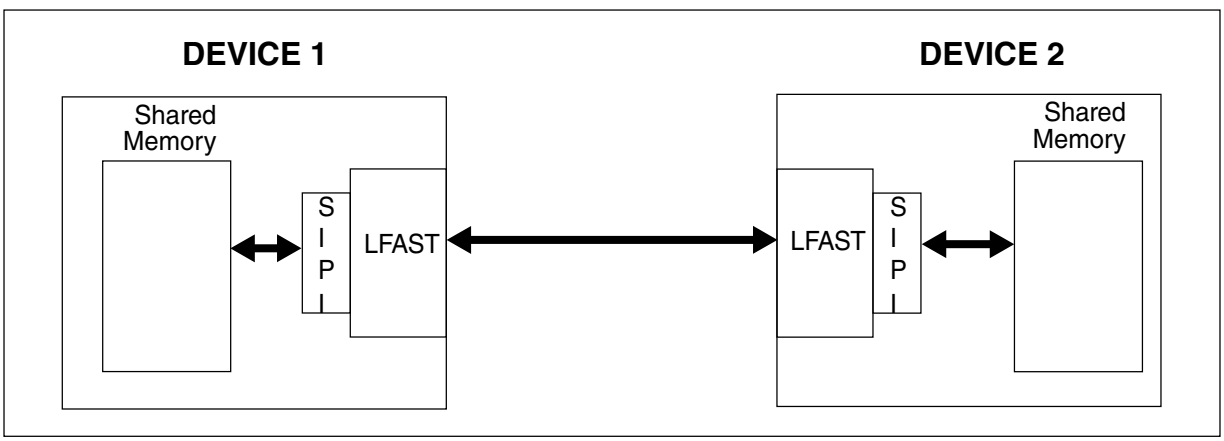


Figure 51-1. Interprocessor communication diagram

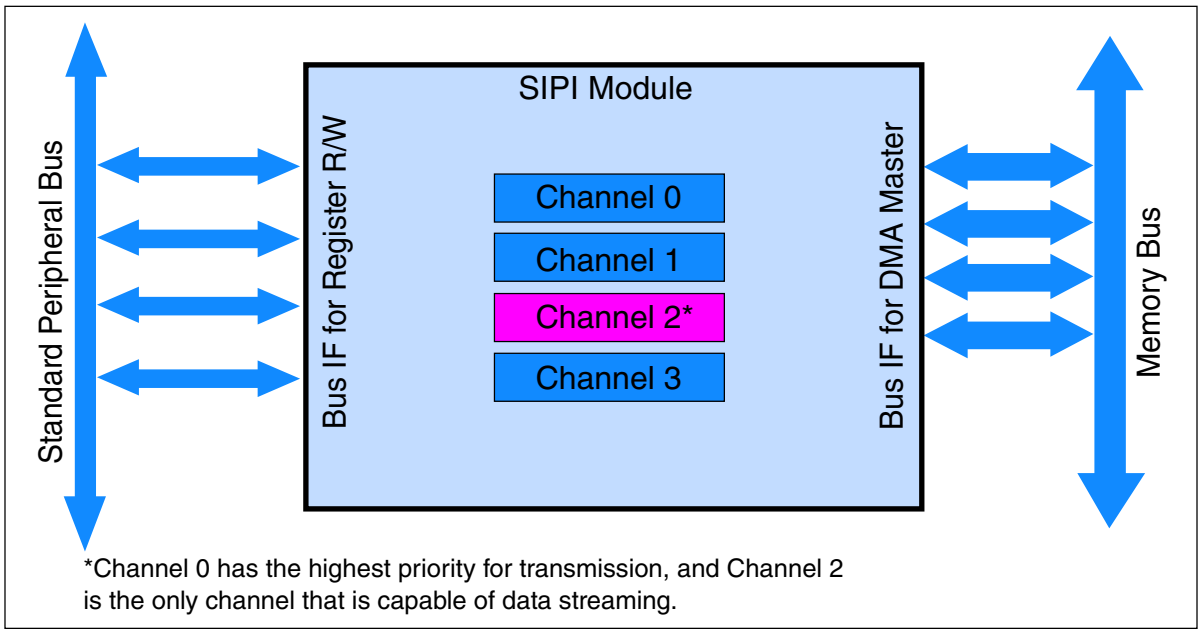
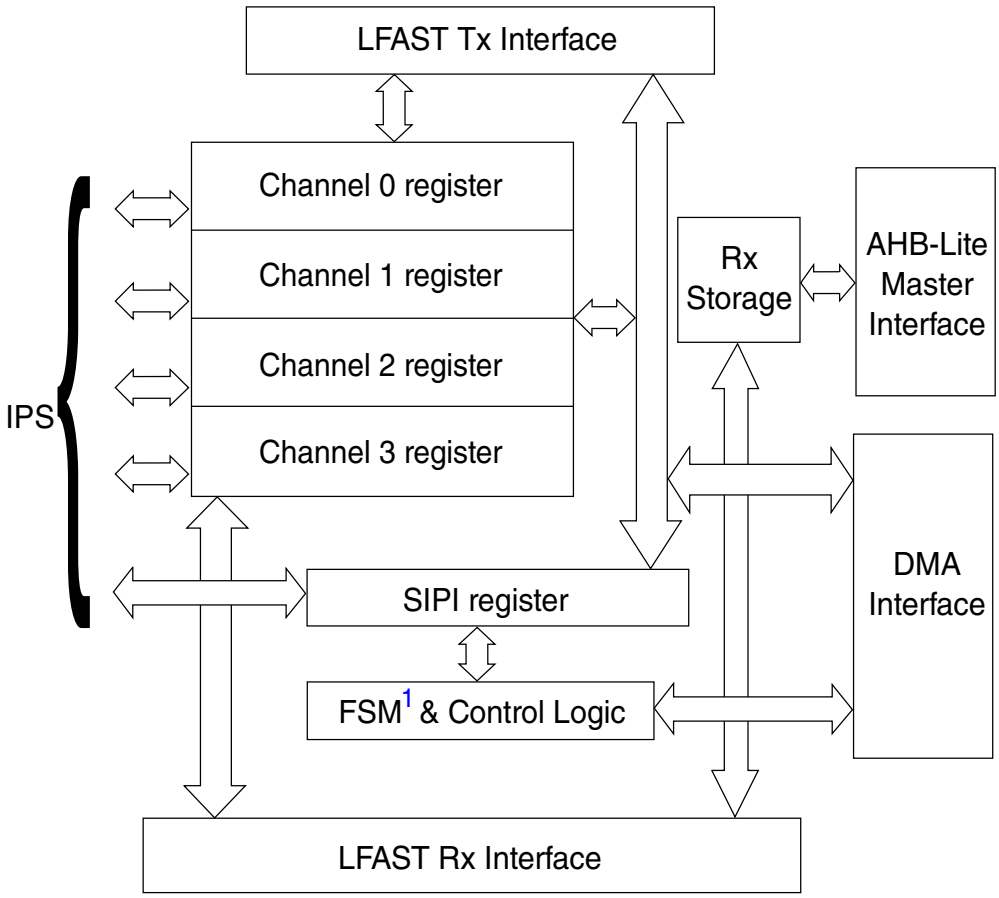


Figure 51-2. SIPI module

51.3 SIPI block diagram



[1] Finite State Machine

Figure 51-3. SIPI block diagram

51.4 Feature description

This section describes the features of the SIPI module.

51.4.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability

- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface which is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator

51.4.2 Standard features

- IPS bus interface (PBRIDGE)
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

51.5 SIPI operation from reset

When the SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

51.6 Functional description

51.6.1 External signals

The SIPI has no chip external signals.

51.6.2 Frame format

All frames have the same general format:

- 16 bit header
- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID. The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

51.6.2.1 Register read request

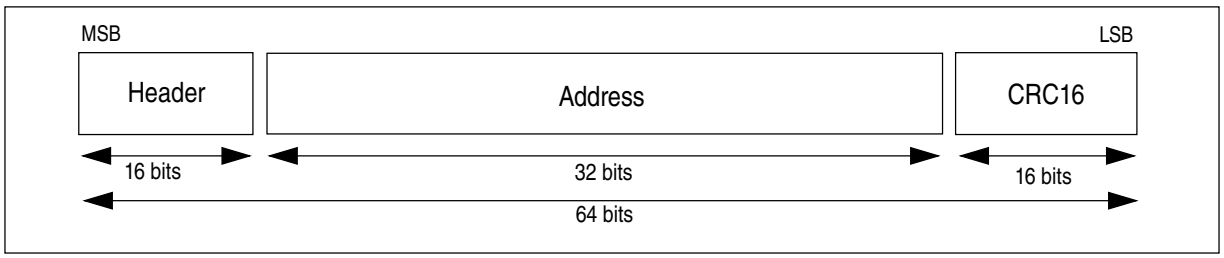


Figure 51-4. SIPI register read request

51.6.2.2 Register read response, ID request response

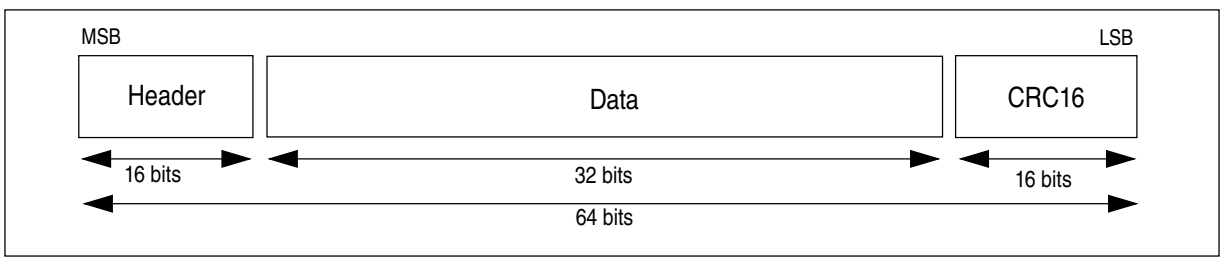


Figure 51-5. SIPI read response

51.6.2.3 Register write request

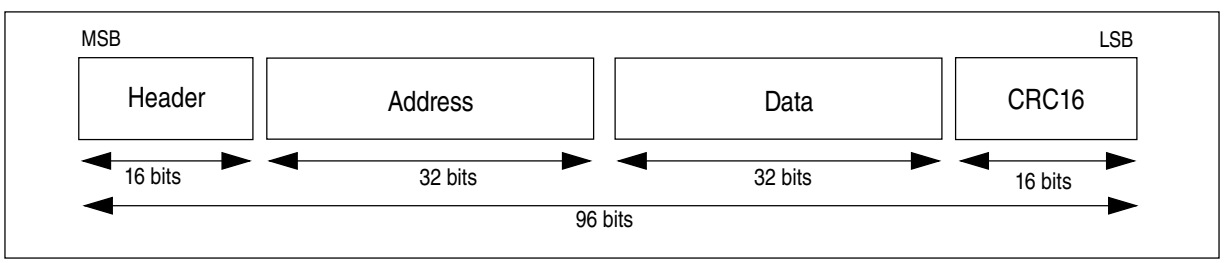


Figure 51-6. SIPI register write request

51.6.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

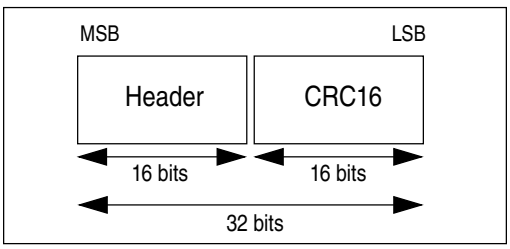


Figure 51-7. SIPI write acknowledge

51.6.2.5 Streaming write request

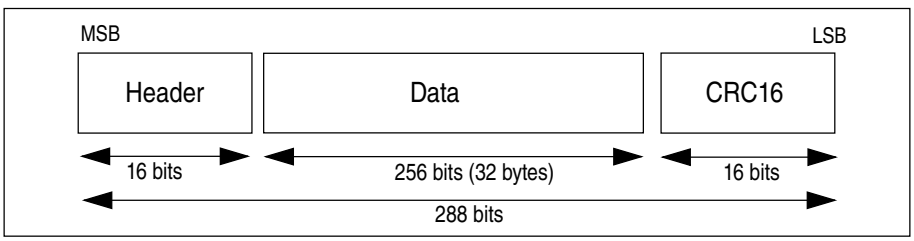


Figure 51-8. Streaming write request format

NOTE

Direct write operations are used to set the streaming address.

Streaming data write is performed using the format shown in Figure 51-8.

51.6.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

51.6.2.6.1 SIPI header coding

Figure 51-9, Table 51-1, Figure 51-10, and Table 51-2 show how the SIPI header bits are coded.

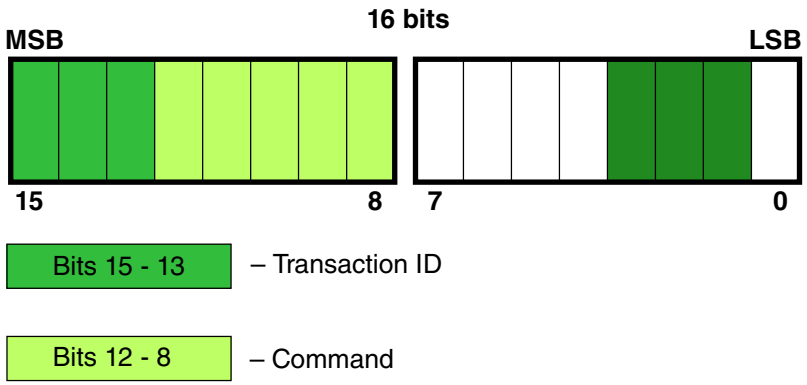


Figure 51-9. SIPI header coding

Table 51-1 shows the command coding for the bits[12:8] of the header.

Table 51-1. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
…	…	…
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
…	…	…
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—
…	…	…
1F	Reserved	—

Figure 51-10 shows the channel number field in the header and Table 51-2 shows the channel number coding.

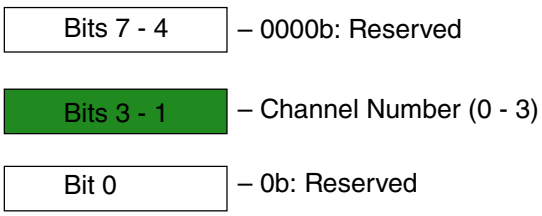


Figure 51-10. SIPI header channel field

Table 51-2. SIPI header channel number coding

SIPI channel name	Channel number coding in header(s)		Comment
	SIPI channel coding select field (SIPI_MCR[CHNSB])		
	Code table I (SIPI_MCR[CHNSB] = 1)	Code table II (SIPI_MCR[CHNSB] = 0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

51.6.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

51.6.2.8 Payload field

Table 51-3 shows the payload sizes of LFAST frames.

Table 51-3. Payload size of LFAST channel frame

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16

Table continues on the next page...

Table 51-3. Payload size of LFAST channel frame (continued)

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 51-4 shows the converted coding of LFAST for a given SIPI code.

Table 51-4. Converted coding of LFAST channel code for SIPI headers

LFAST Code	SIPI Code	Channel (hex)[1]
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by SIPI)
1001b	001b	F (not used by SIPI)
1010b	010b	G (not used by SIPI)
1011b	011b	H (not used by SIPI)

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

51.6.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). MSB is sent first in the data stream.

51.6.2.9.1 CRC field examples

51.6.2.9.1.1 Example 1 – 32 bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

51.6.2.9.1.2 Example 2 – 32 bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

51.6.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

51.7 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

51.7.1 Read transfer

A Read transfer can be of two types:

- Read Request Transfer (at initiator node)
- Read Response Transfer (at target node)

51.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in Figure 51-11.

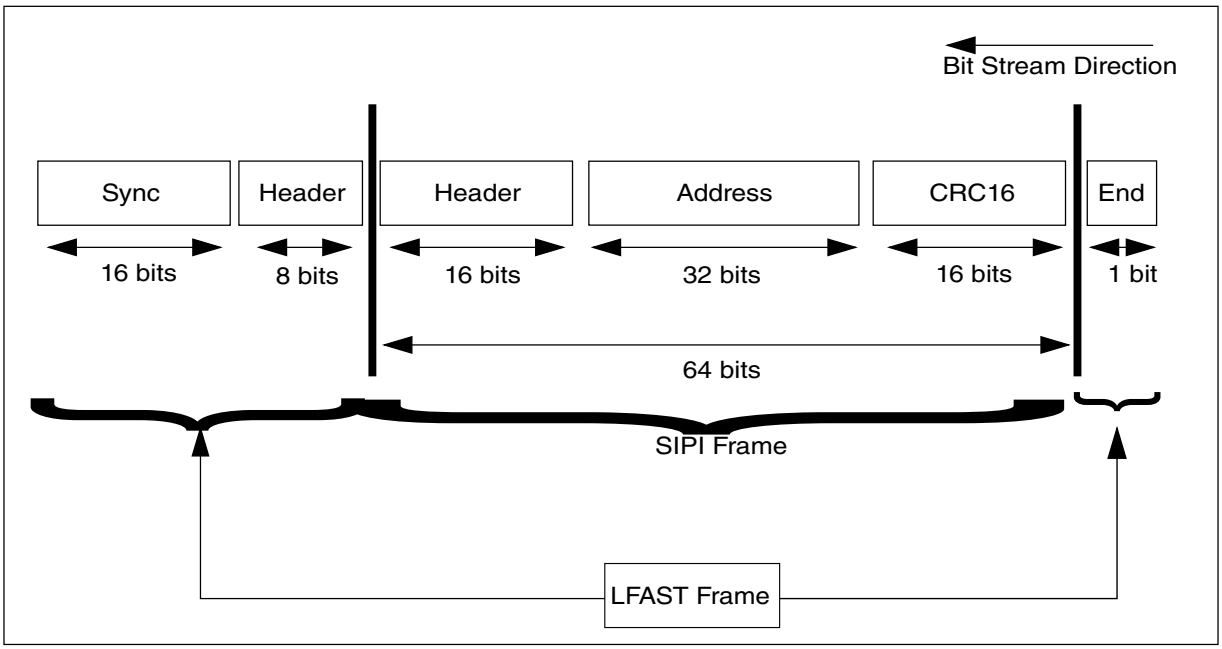


Figure 51-11. Read request transfer

51.7.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see Figure 51-12). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see Figure 51-13).

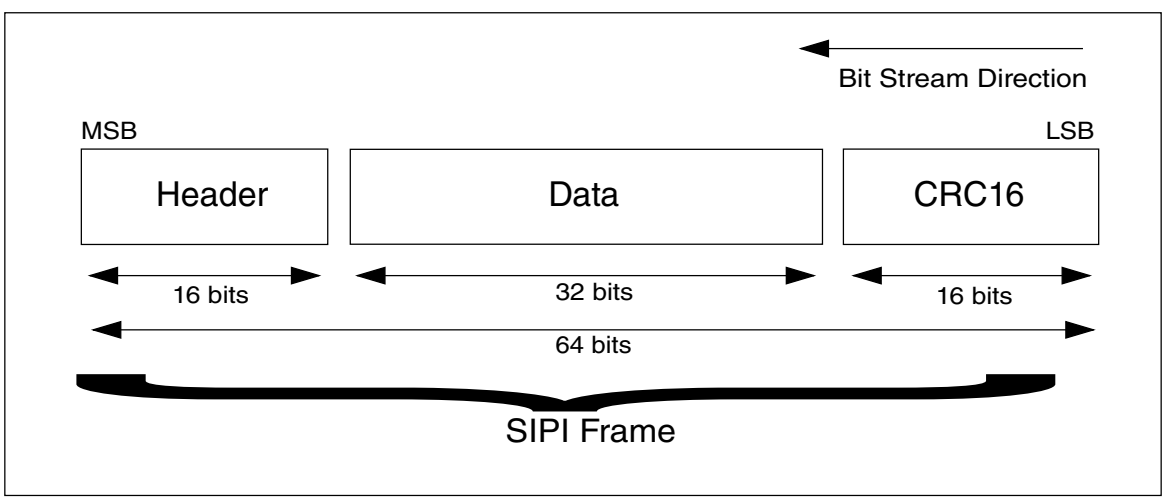


Figure 51-12. Read answer transfer

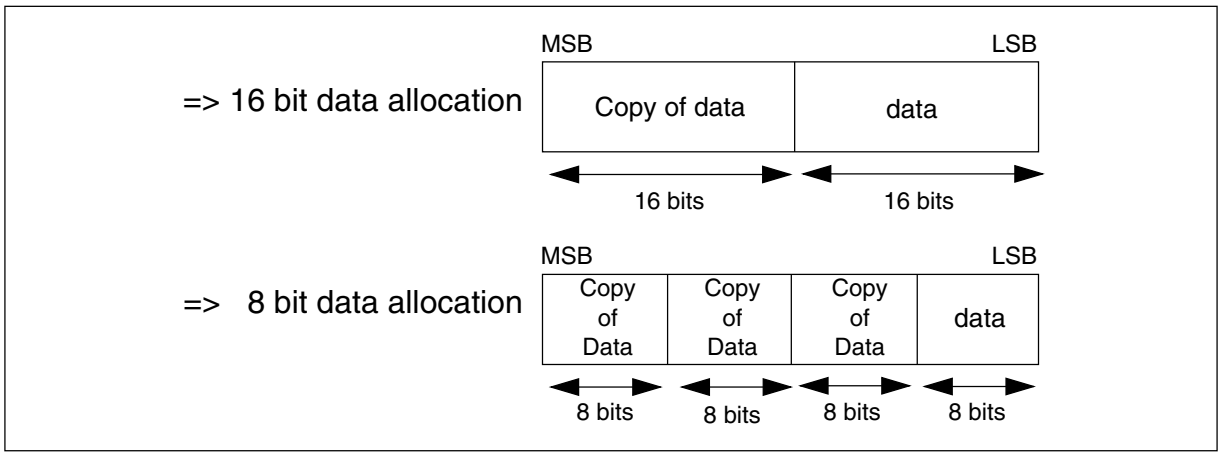


Figure 51-13. Data allocation

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as halfword enable.
  - Bit[0]:
    - 0 - halfword 1 enabled (MSB)
    - 1 - halfword 0 enabled (LSB)

51.7.3 Register Write transfer

Register Write transfer can be of two types:

- Normal write transfer - channels 0, 1, 2 and 3
- Streaming data transfer - channel 2 only

A Register Write transfer can be done through Normal Write transfer on channels 0, 1, 2 and 3 (see Normal write transfer) as shown in Figure 51-14.

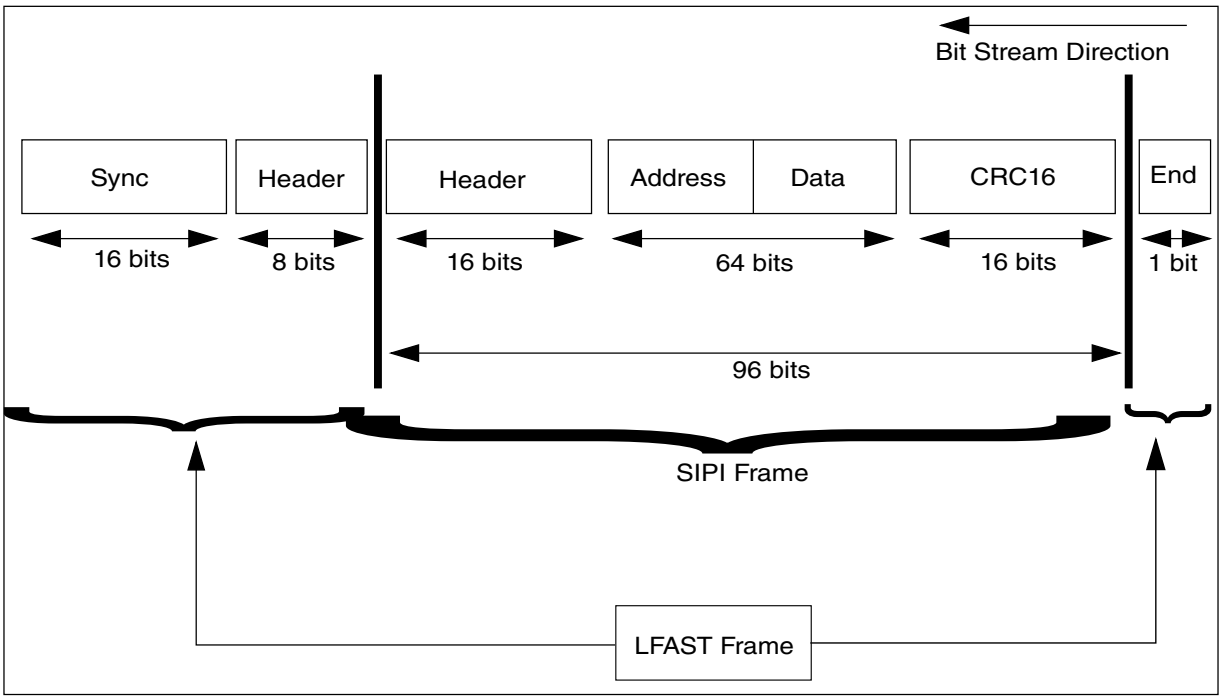


Figure 51-14. Register write transfer (showing LFAST frame encapsulation)

51.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and CRC as shown in Figure 51-15.

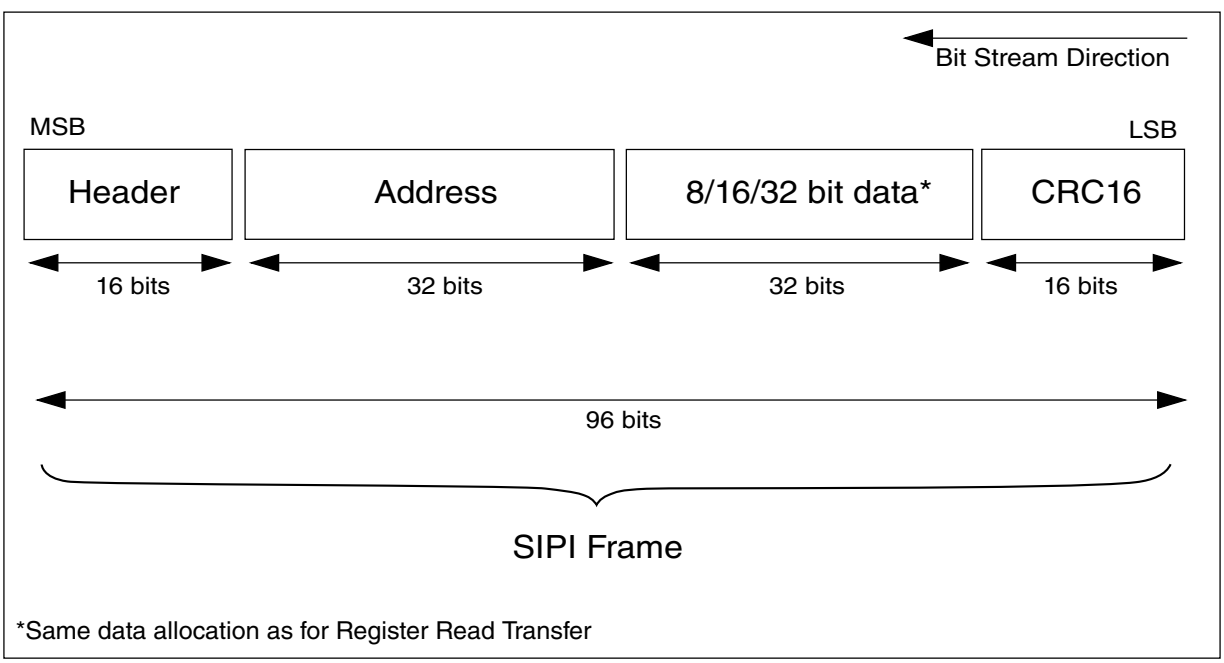


Figure 51-15. Write transfer (LFAST frame encapsulation is not shown)

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
  - Bit[0]:
    - 0 - half-word 1 enabled (MSB)
    - 1 - half-word 0 enabled (LSB)

51.7.3.2 Streaming write transfer

Streaming write transfer has 256 bits of payload.

51.7.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. The SIPI Maximum Count Register (SIPI_MAXCR) and SIPI Address Reload Register (SIPI_ARR) are written before the SIPI Address Count Register (SIPI_ACR). This is to avoid unwanted behavior of the SIPI attempting to access non-shared memory. This is only true for the very first streaming command, subsequent streaming command(s) may not need to write the SIPI_MAXCR and SIPI_ARR (see SIPI Max Count Register (SIPI_MAXCR), SIPI Address Count Register (SIPI_ACR) and SIPI Address Reload Register (SIPI_ARR)).

51.7.3.2.2 Streaming transfer with data

Figure 51-16 shows the packet structure of the streaming transfer containing data.

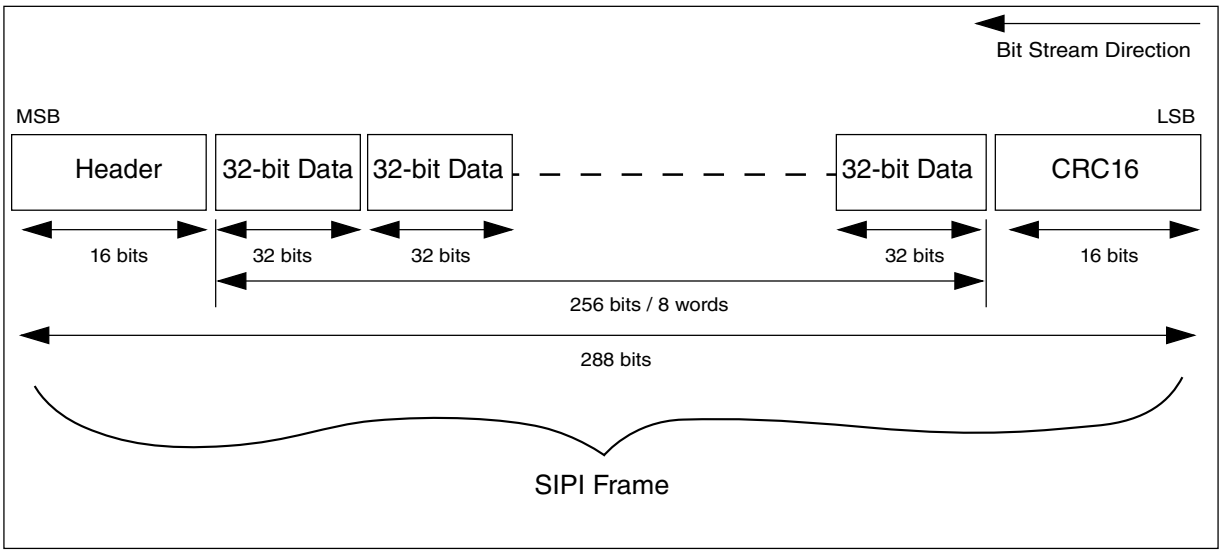


Figure 51-16. Streaming transfer with data

51.7.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see Write Aknowledge transfer. The CRC bits are calculated on the header field.

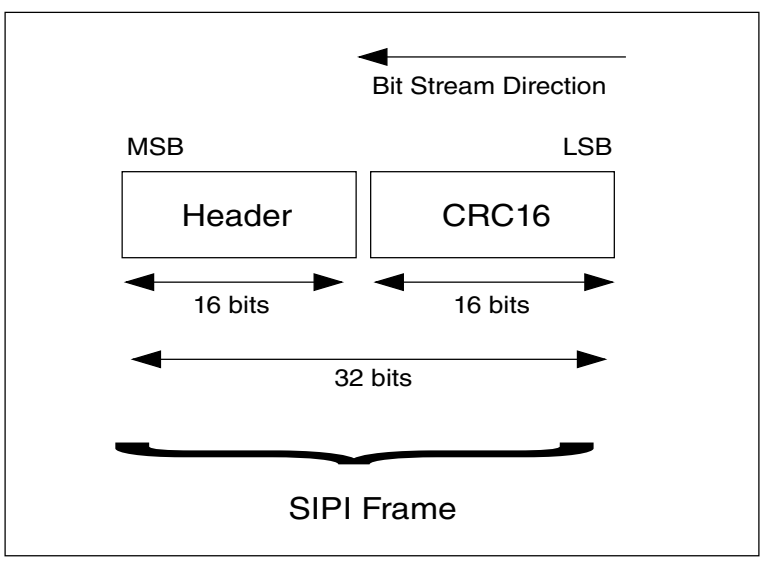


Figure 51-17. Write Aknowledge transfer (LFAST encapsulation is not shown)

51.7.5 ID request response

51.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in Figure 51-18.

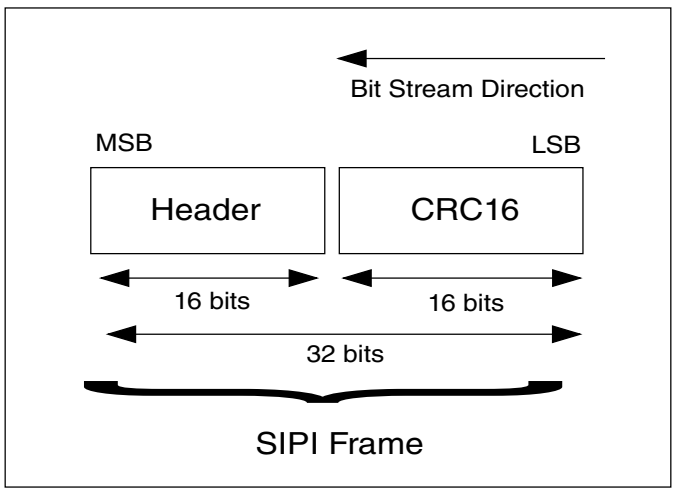


Figure 51-18. ID request transfer (LFAST encapsulation is not shown)

51.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in Figure 51-19.

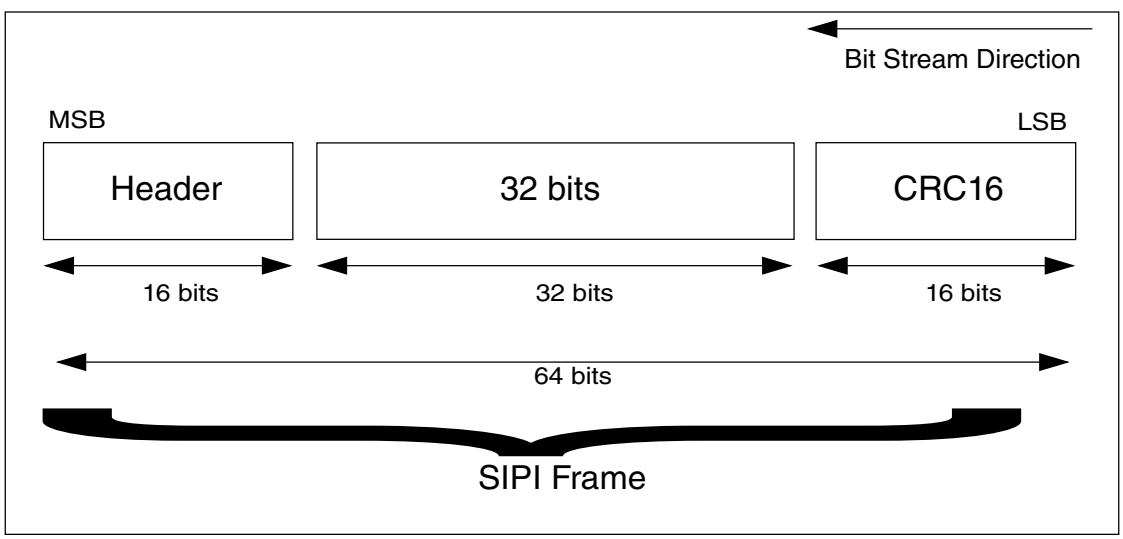


Figure 51-19. ID request response transfer

Note

The ID request response contains the value of the CHIP ID in place of data.

51.8 Transfer API and flow charts

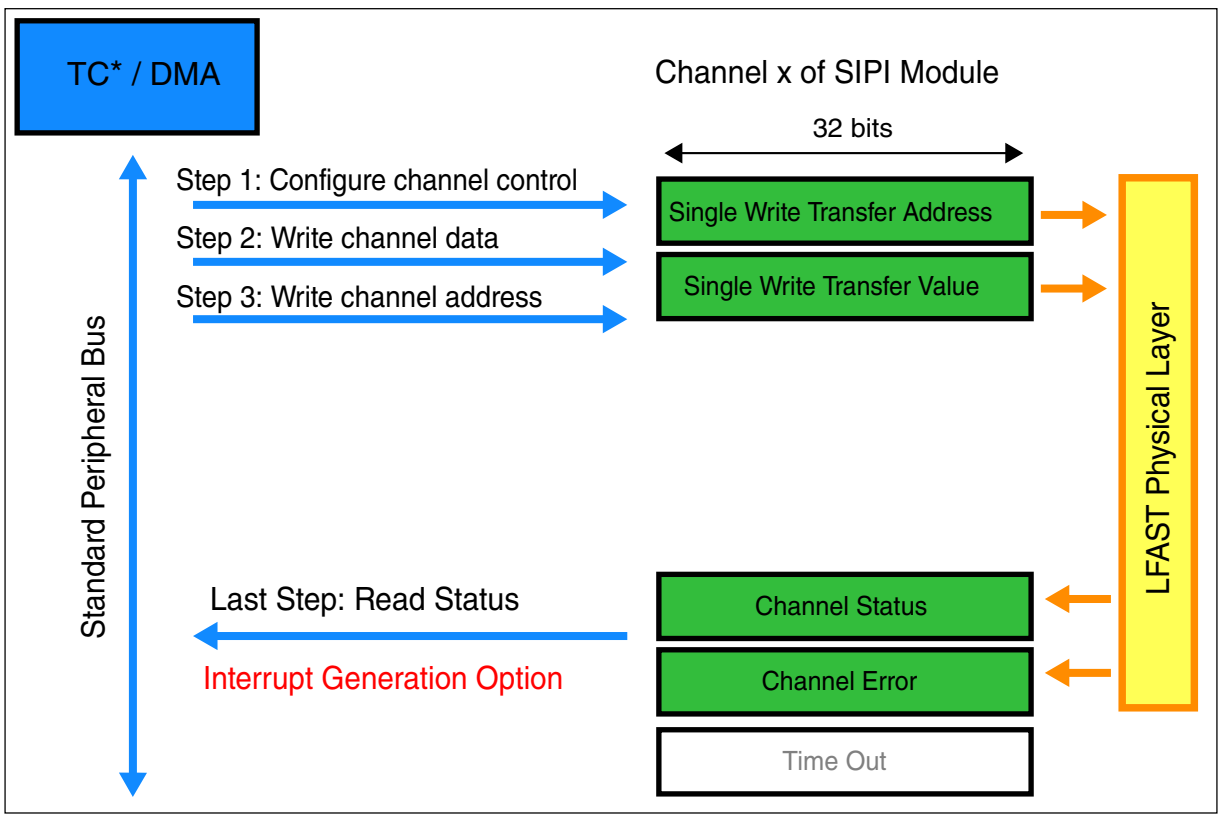


Figure 51-20. SIPI single register write API

Implement the following steps to generate a single Write Transfer Request (Figure 51-20):

1. Configure data and SIPI_CCR*n* at the initiator node.
2. Configure SIPI_CAR*n* at the initiator node.

   As soon as the channel address register (SIPI_CAR*n*) is written and if CCR*n*[CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.
3. Software polls the status register bits (SIPI_CSR*n*) to determine when the request completes, SIPI_CSR*n*[ACKR] = 1. An interrupt will be generated if the corresponding SIPI_CIR*n*[ACKIE] = 1.

On a single Write transfer request reception (Figure 51-21):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process in completed, the target node will generate an acknowledge frame and send it back to the LFAST.

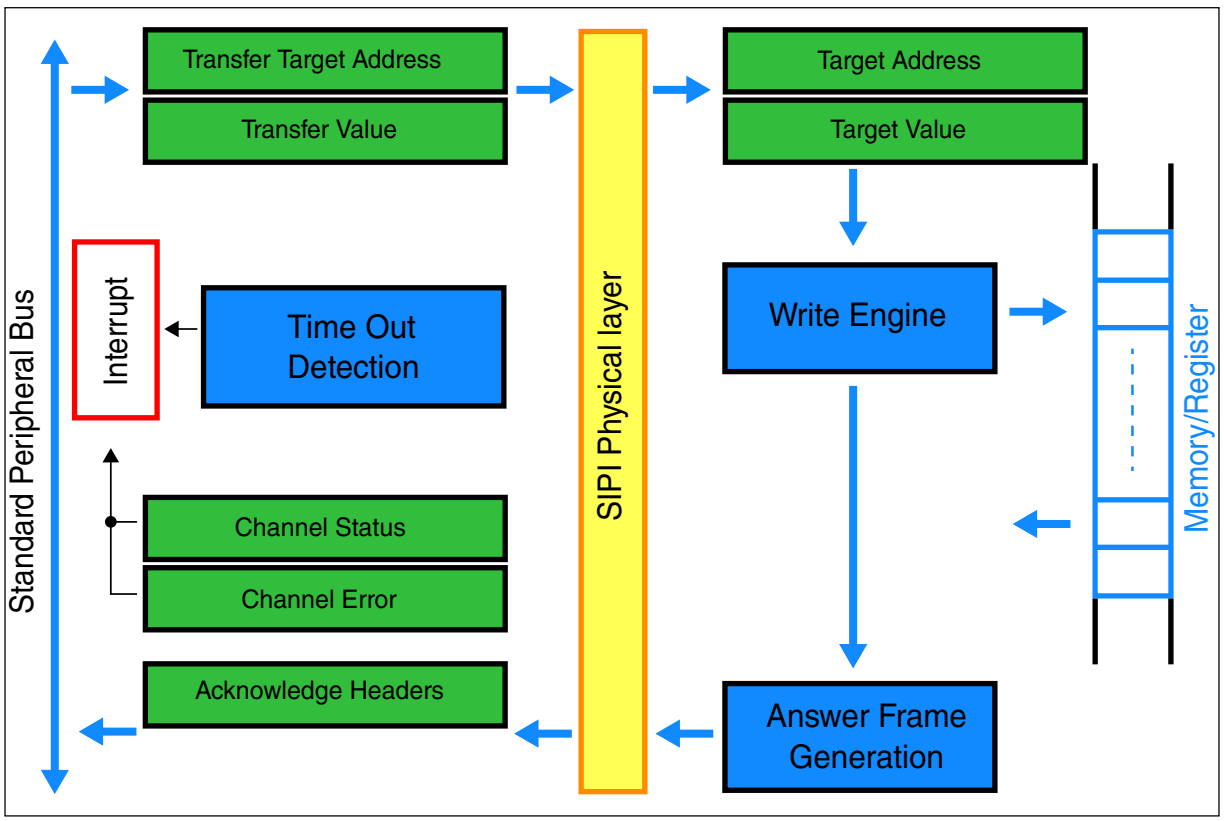


Figure 51-21. SIPI Single Register Write API – Flow Chart

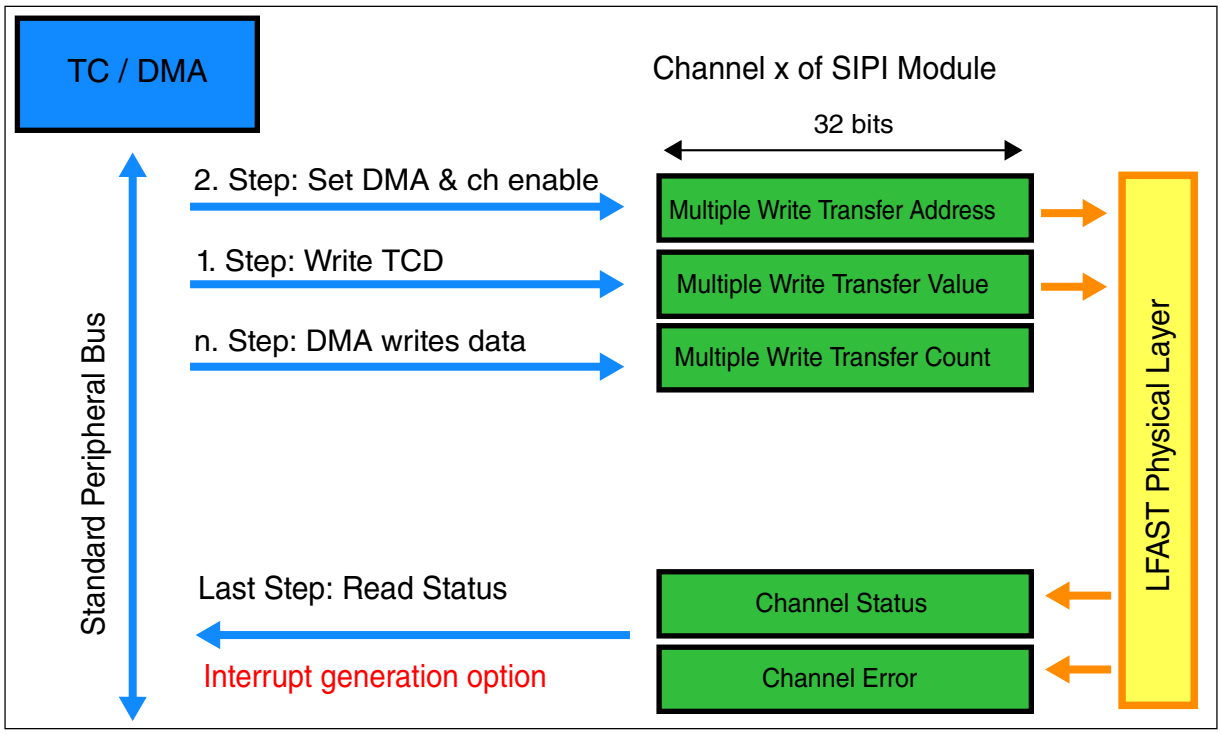


Figure 51-22. SIPI Multiple Register Write API

For multiple write transfer request generation (Figure 51-22):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write SIPI_CCR*n*[CHEN] = 1 and SIPI_CCR*n*[DAN] = 1.
3. SIPI will start copying data into the SIPI_CDR*n* through its DMA interface, depending on the transfer count and data registers size. SIPI_CDR2_0 should be written with the MSB.
4. When the copying process is completer, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
5. Software should poll the SIPI_CSR*n* status register bits to determine if the request has completed. If SIPI_CSR*n*[ACKR] = 1, an interrupt will be generated (if the corresponding SIPI_CIR*n*[ACKIE] = 1).
6. If SIPI_CCR*n*[DEN] = 1, the SIPI request will transfer to the DMA controller. If not, the state machine go idle.
7. Steps 4–7 will be repeated.

On Multiple Write transfer request reception (Figure 51-22):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, target node will generate an acknowledge frame and send it back to the LFAST.

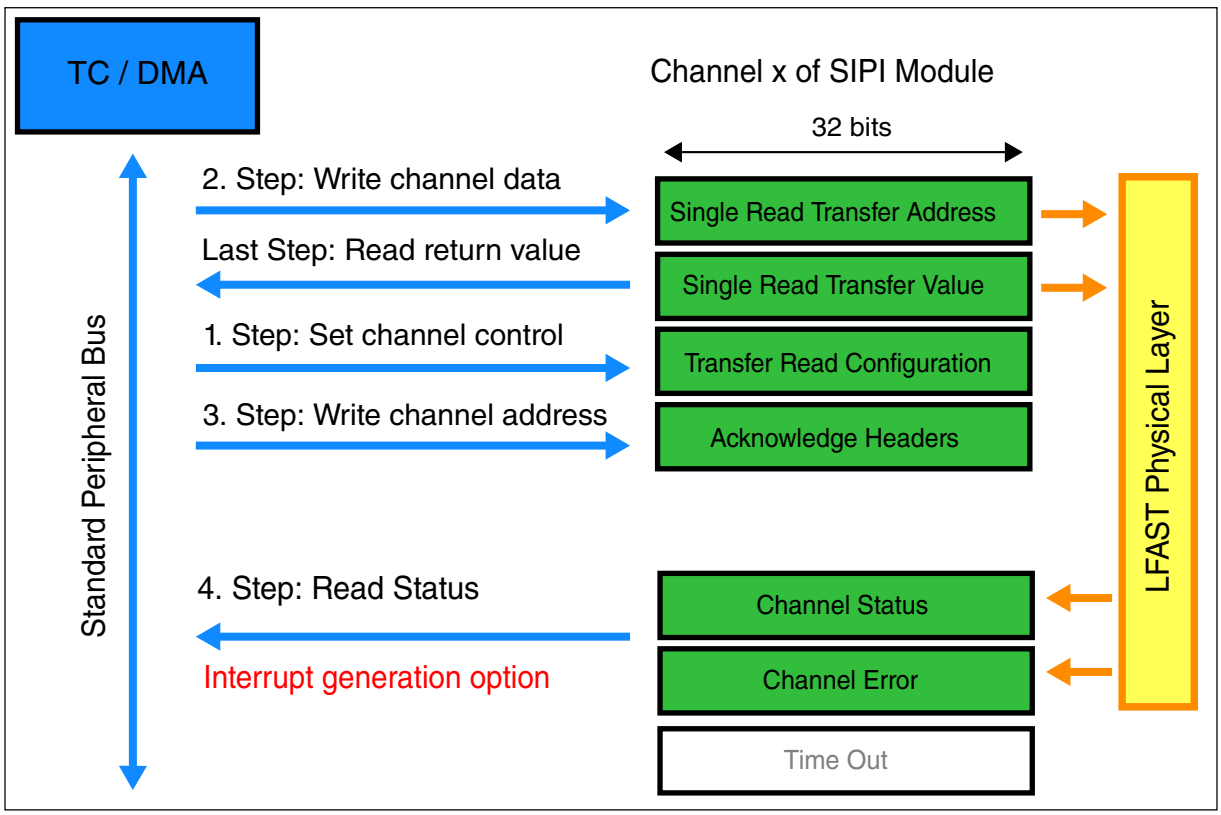


Figure 51-23. SIPI Single Register Read API

For Single Read Transfer Request generation (Figure 51-23):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing CAR*n*.
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1, an interrupt will be generated (if SIPI_CIR*n*[RAIE] = 1). If SIPI_CSR*n*[RAR] does not set then SIPI_ERR[TOE*n*] = 1 which indicates a timeout has occured.
4. If SIPI_CSR*n*[RAR] = 1 then software can read the data register, or can other necessary action if SIPI_CSR*n*[RAR] = 0.

On Single Read transfer request reception (Figure 51-23):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process in completed, target node will send read response back to LFAST.

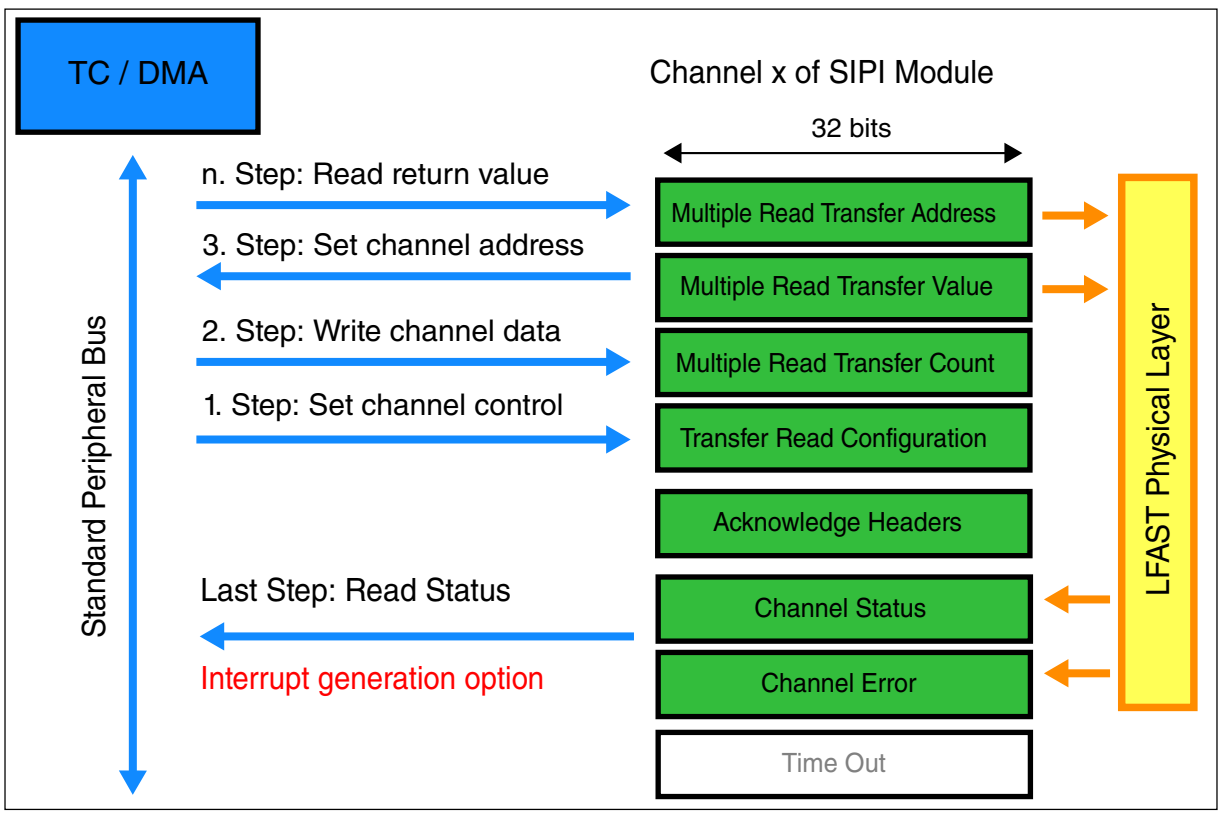


Figure 51-24. SIPI multiple register read API

For multiple read transfer request generation (Figure 51-24):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing SIPI_CAR*n*.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then starts transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1 and SIPI_CIR*n*[RAIE] = 1 an interrupt will be generated. If SIPI_CSR*n*[RAR] does not set, then SIPI_ERR[TOE*n*] = 1 which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception (Figure 51-24):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

Figure 51-25, Figure 51-26, Figure 51-27 and Figure 51-28 show the SIPI data streaming of data.

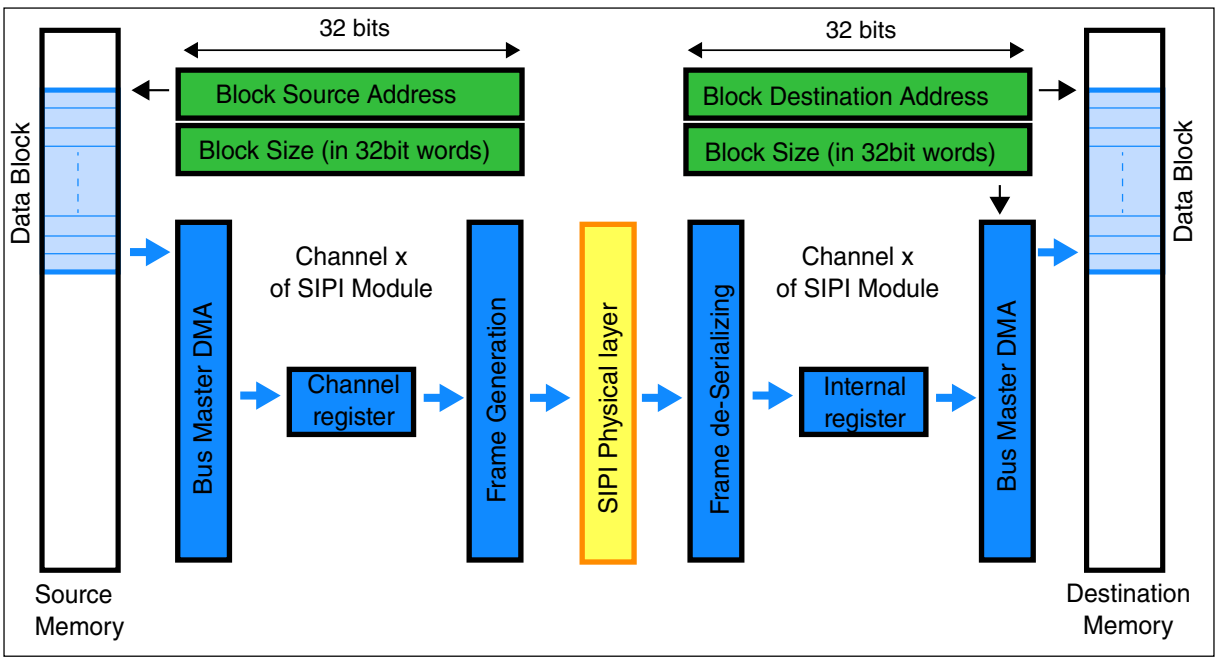


Figure 51-25. SIPI data stream model

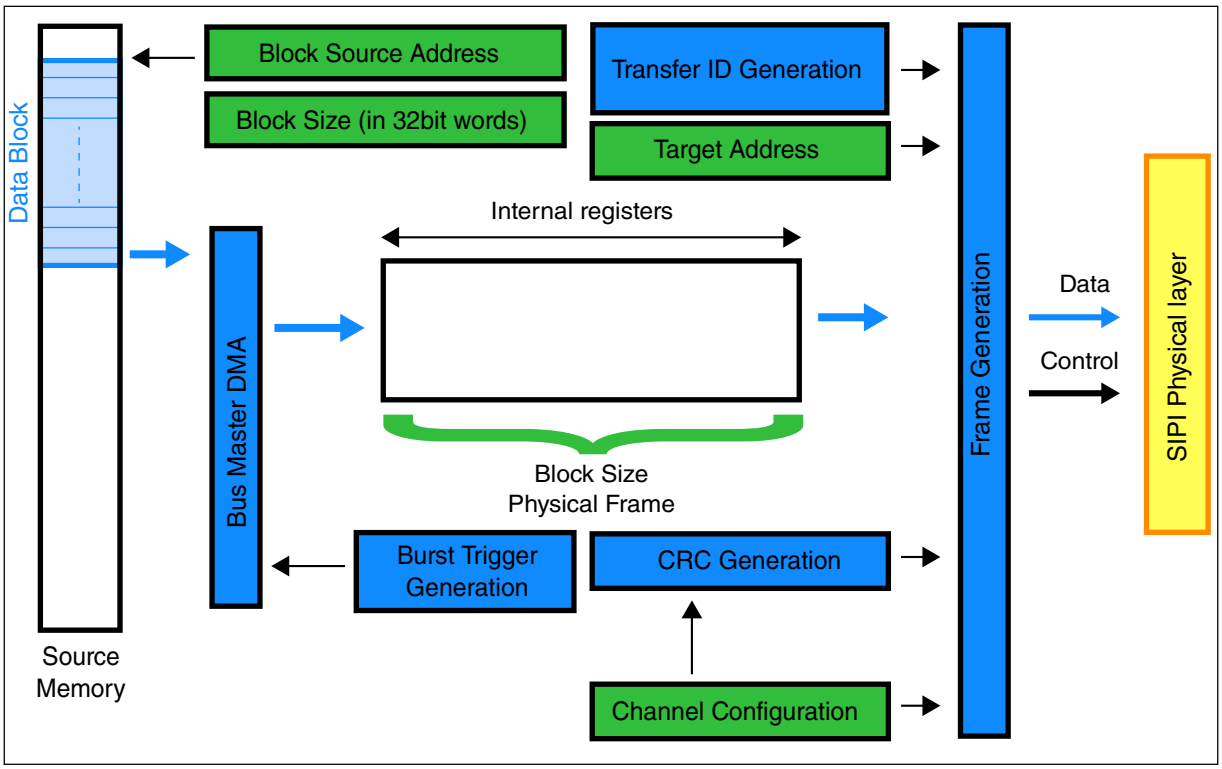


Figure 51-26. SIPI data stream model - Initiator

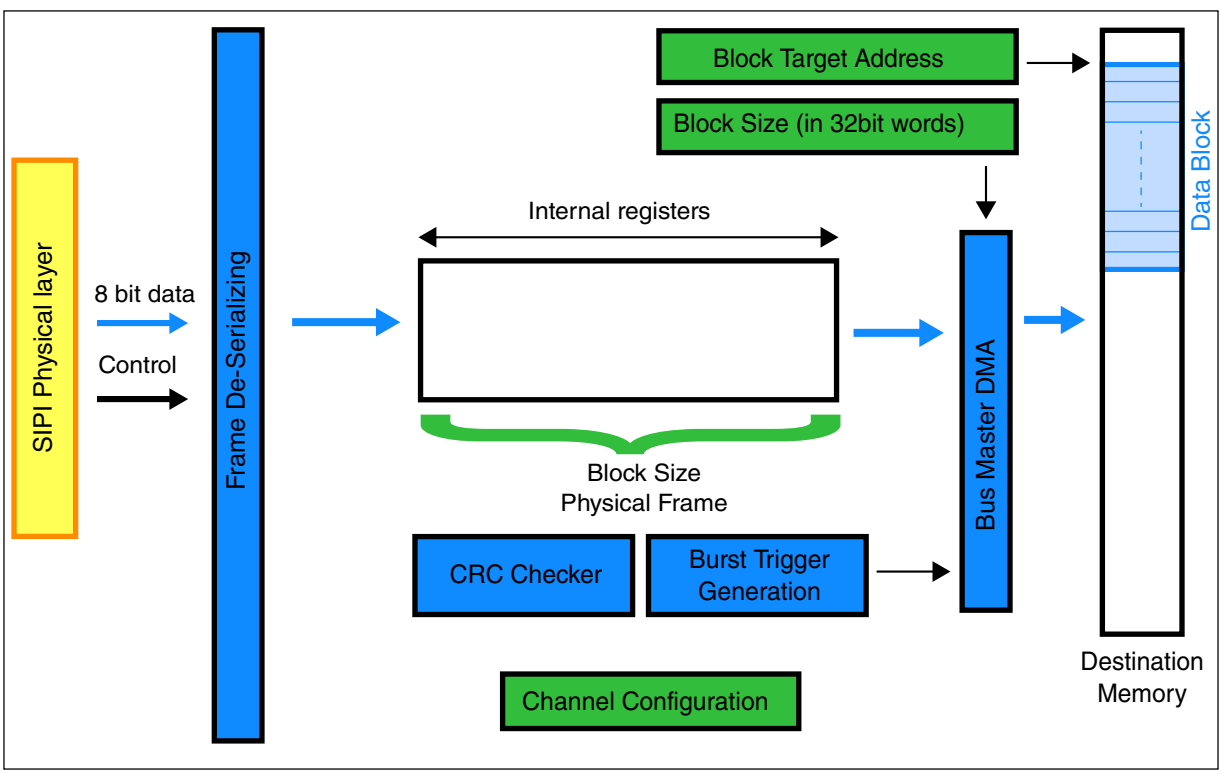


Figure 51-27. SIPI data stream model - Target

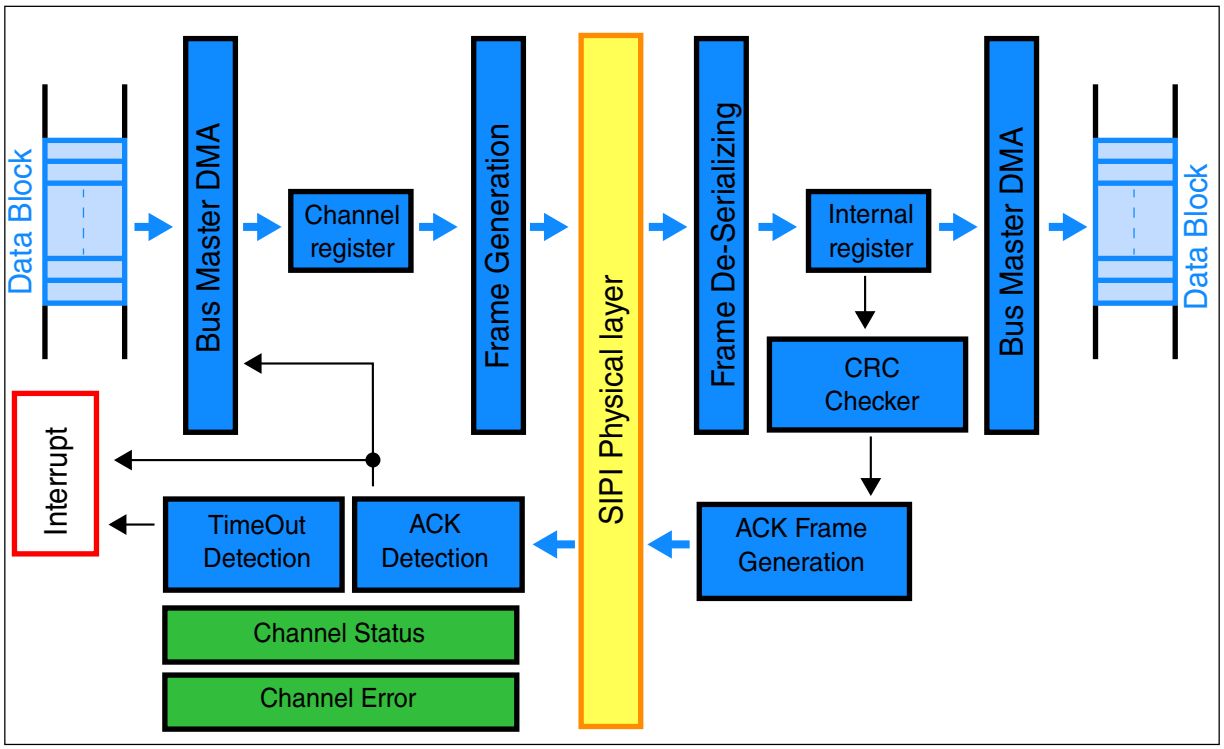


Figure 51-28. SIPI data stream with acknowledge trigger

51.9 DMA programming sequence

The DMA programming sequence is as follows:

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. If SIPI has received a read response, and error flags are not set on the LFAST Rx port, SIPI will assert ipd_request. In case of error, move to Step 8.
6. As soon as ipd_request is asserted, DMA transfer starts between the SIPI registers and RAM.
7. If DMA transfer minor loop/major loop is complete, and the request is negated, ipd_request is negated.
8. If an error occurs:

- The SIPI generates an interrupt
- Software takes control of the SIPI

9. Steps 5 – 8 are repeated until the transfer is finished.

51.10 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in module disable mode, which helps reduce power consumption.

51.10.1 Initialization mode

To enter Initialization mode software writes SIPI_MCR[INIT]=1 SIPI_MCR[MOEN] must be set before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT]=0 (see SIPI Module Configuration Register (SIPI_MCR)).

All message transfers are stopped when in Initialization mode. If software invokes Initialization mode when a bus transfer is in progress, the transfer will be aborted immediately even if the transfer has not completed.

Note

It is recommended that software checks the state of SIPI (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

51.10.2 Normal mode

Once software has completed initialization of SIPI (SIPI_MCR[INIT]=1), it can enter Normal mode by writing SIPI_MCR[INIT]=0. SIPI needs to be in Normal mode for all data transfers (see SIPI Module Configuration Register (SIPI_MCR)).

Note

SIPI must be enabled before a attempting to write SIPI_MCR[INIT] (SIPI_MCR[MOEN]=1).

51.10.3 Module Disable (MD)

MD mode in the SIPI is used to help reduce power consumption. By default, SIPI is in Disable mode, SIPI_MCR[MOEN]=0, and is exited by writing SIPI_MCR[MOEN]=1 (see SIPI Module Configuration Register (SIPI_MCR)). All the activities on the SIPI Tx and Rx ports are immediately stopped in Module Disable mode.

51.11 Errors

This section describes the potential errors that can occur during SIPI operation.

51.11.1 Timeout error

A timeout error is generated at the initiator node when the acknowledge/response is not received within the time configured in the corresponding CTOR*n*[TOR] field setting (see SIPI Channel Timeout Register 0 (SIPI_CTOR0), SIPI Channel Timeout Register 1 (SIPI_CTOR1), SIPI Channel Timeout Register 2 (SIPI_CTOR2) and SIPI Channel Timeout Register 3 (SIPI_CTOR3)). A timeout error is indicated when ERR[TOE*n*]=1 (see SIPI Error Register (SIPI_ERR)).

Note

SIPI should not drop the response even after a timeout occurs. Software will poll both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may then be discarded.

51.11.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match the calculated CRC, and the SR[GCRCE] is set (see SIPI Status Register (SIPI_SR)). An interrupt will be asserted if MCR[CRCIE]=1 (see SIPI Module Configuration Register (SIPI_MCR)).

Note

The target node will not send an acknowledge to the initiator node when a CRC error is generated. An interrupt will be generated on the target side if the corresponding interrupt

enable bit is set. The initiator node will detect a timeout and take necessary action.

51.11.3 Maximum count reached error

The maximum count reached error is only generated at the target node. It is generated when the value of the SIPI_ACR is equal to the SIPI_MAXCR (see SIPI Max Count Register (SIPI_MAXCR) and SIPI Address Count Register (SIPI_ACR)). When the maximum count is reached, SIPI_SR[MCR] = 1 (see SIPI Status Register (SIPI_SR)). An interrupt will be generated if SIPI_MCR[MCRIE] = 1 (see SIPI Module Configuration Register (SIPI_MCR)).

51.11.4 Transaction ID error

The Transaction ID (TID) error is always generated at the initiator node only. It is generated when header bits 15–13 do not match SIPI_CSR*n*[TID] (transaction ID bits, see SIPI Channel Status Register 0 (SIPI_CSR0), SIPI Channel Status Register 1 (SIPI_CSR1), SIPI Channel Status Register 2 (SIPI_CSR2), and SIPI Channel Status Register 3 (SIPI_CSR3)). SIPI_CSR*n*[TIDE] = 1 when a TID error is detected, and an interrupt will be generated if SIPI_CIR*n*[TIDIE] = 1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

51.11.5 Acknowledge error

An incorrect acknowledge is received only from the initiator. When the acknowledge received is incorrect, SIPI_ERR[ACKR*n*] will be set (see SIPI Error Register (SIPI_ERR)). An interrupt will be generated if SIPI_CIR*n*[WAIE]=1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

51.12 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

1) The CRC seed is initialized by FFFF_FFFFh.

2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h.

4) Address will be sent as 22CC_4488h.

5) Data will be sent as FF77_BB33h.

51.13 Interrupt logic

A description of the interrupt logic can be found in the following figure.

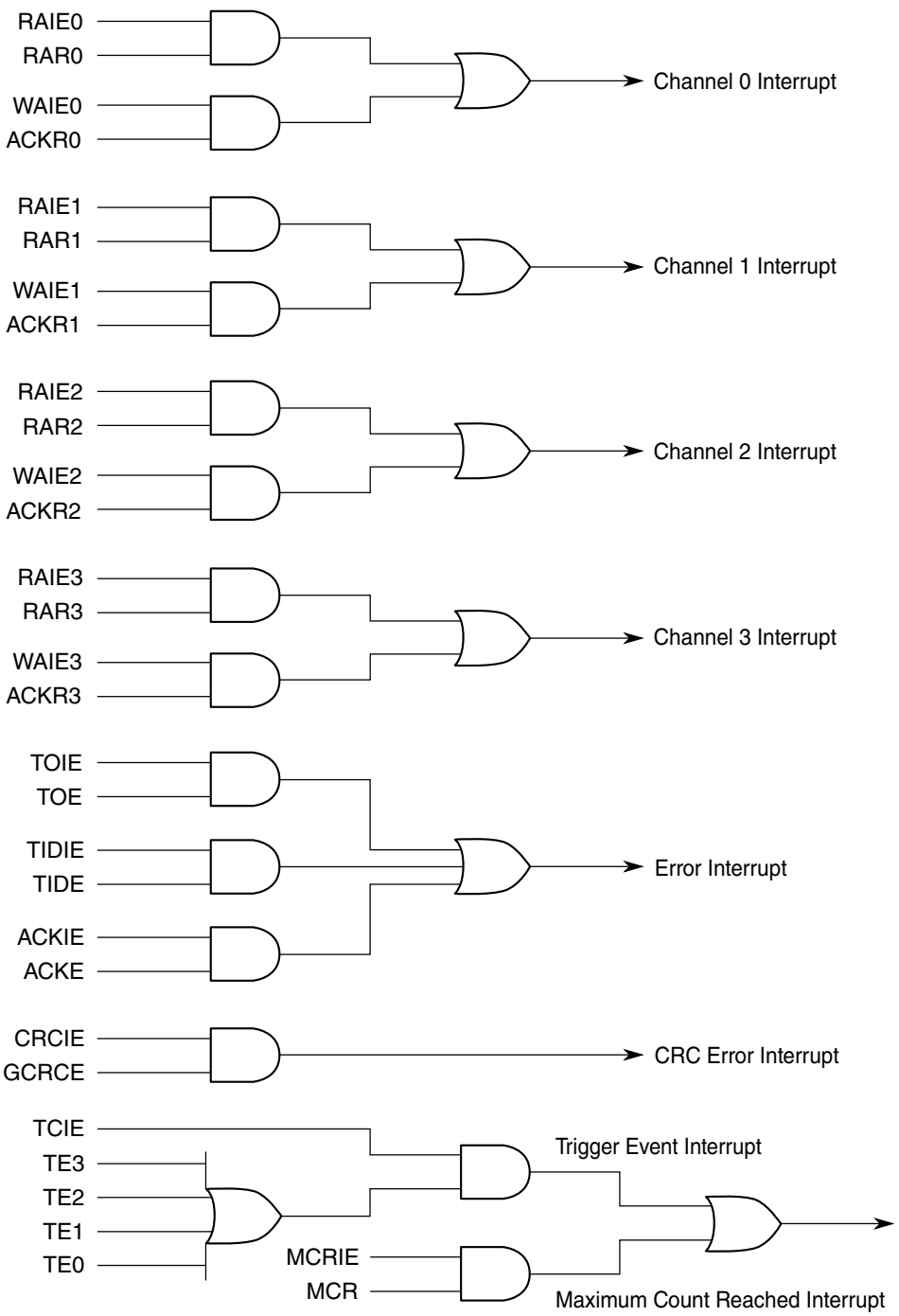


Figure 51-29. Interrupt description

51.14 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

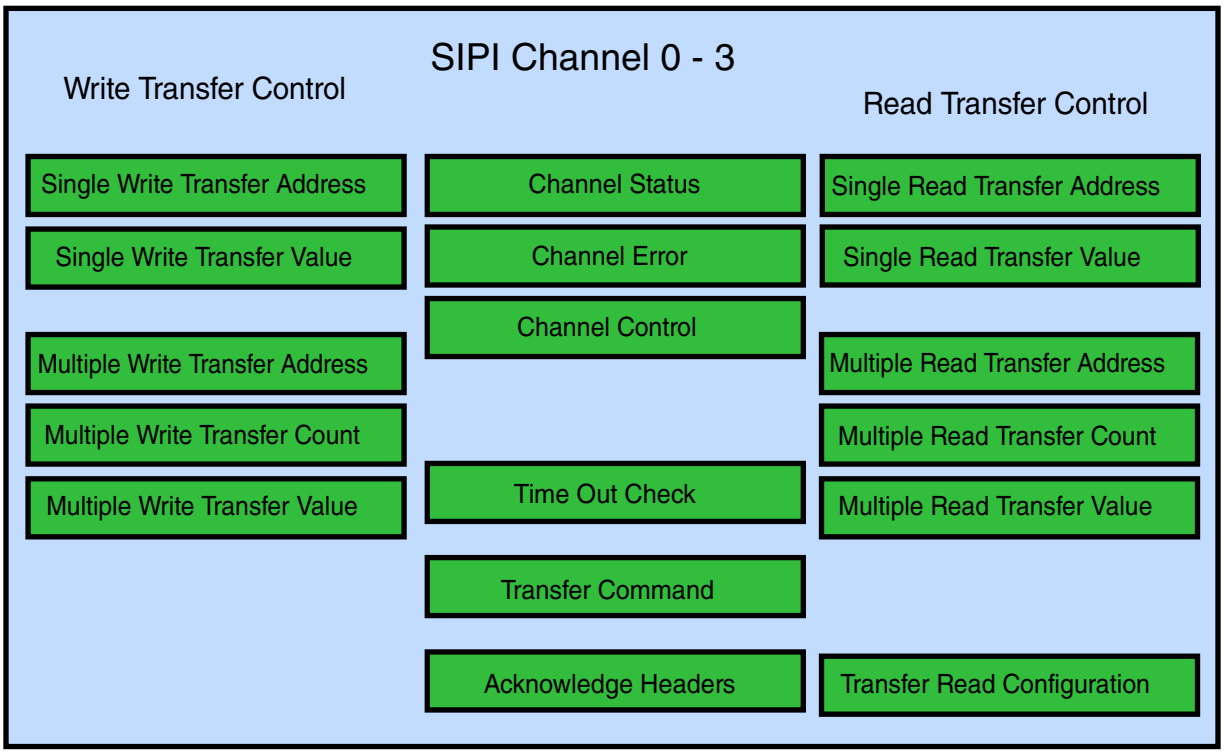


Figure 51-30. SIPI control and status overview – Register transfer

51.15 Memory map and register definition

SIPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SIPI Channel Control Register 0 (SIPI_CCR0)	32	R/W	0000_0000h	51.15.1/ 2213
4	SIPI Channel Status Register 0 (SIPI_CSR0)	32	R	0000_0000h	51.15.2/ 2216
C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	32	R/W	0000_0000h	51.15.3/ 2217
10	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	32	R/W	0000_00FFh	51.15.4/ 2218
14	SIPI Channel CRC Register 0 (SIPI_CCRC0)	32	R	0000_0000h	51.15.5/ 2219
18	SIPI Channel Address Register 0 (SIPI_CAR0)	32	R/W	0000_0000h	51.15.6/ 2219
1C	SIPI Channel Data Register 0 (SIPI_CDR0)	32	R/W	0000_0000h	51.15.7/ 2220
20	SIPI Channel Control Register 1 (SIPI_CCR1)	32	R/W	0000_0000h	51.15.8/ 2220

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	SIPI Channel Status Register 1 (SIPI_CSR1)	32	R	0000_0000h	51.15.9/ 2223
2C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	32	R/W	0000_0000h	51.15.10/ 2225
30	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	32	R/W	0000_00FFh	51.15.11/ 2226
34	SIPI Channel CRC Register 1 (SIPI_CCRC1)	32	R	0000_0000h	51.15.12/ 2227
38	SIPI Channel Address Register 1 (SIPI_CAR1)	32	R/W	0000_0000h	51.15.13/ 2227
3C	SIPI Channel Data Register 1 (SIPI_CDR1)	32	R/W	0000_0000h	51.15.14/ 2228
40	SIPI Channel Control Register 2 (SIPI_CCR2)	32	R/W	0000_0000h	51.15.15/ 2228
44	SIPI Channel Status Register 2 (SIPI_CSR2)	32	R	0000_0000h	51.15.16/ 2231
4C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	32	R/W	0000_0000h	51.15.17/ 2233
50	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	32	R/W	0000_00FFh	51.15.18/ 2234
54	SIPI Channel CRC Register 2 (SIPI_CCRC2)	32	R	0000_0000h	51.15.19/ 2235
58	SIPI Channel Address Register 2 (SIPI_CAR2)	32	R/W	0000_0000h	51.15.20/ 2235
5C	SIPI Channel Data Register 2 (SIPI_CDR2_0)	32	R/W	0000_0000h	51.15.21/ 2236
60	SIPI Channel Data Register 2 (SIPI_CDR2_1)	32	R/W	0000_0000h	51.15.21/ 2236
64	SIPI Channel Data Register 2 (SIPI_CDR2_2)	32	R/W	0000_0000h	51.15.21/ 2236
68	SIPI Channel Data Register 2 (SIPI_CDR2_3)	32	R/W	0000_0000h	51.15.21/ 2236
6C	SIPI Channel Data Register 2 (SIPI_CDR2_4)	32	R/W	0000_0000h	51.15.21/ 2236
70	SIPI Channel Data Register 2 (SIPI_CDR2_5)	32	R/W	0000_0000h	51.15.21/ 2236
74	SIPI Channel Data Register 2 (SIPI_CDR2_6)	32	R/W	0000_0000h	51.15.21/ 2236
78	SIPI Channel Data Register 2 (SIPI_CDR2_7)	32	R/W	0000_0000h	51.15.21/ 2236
7C	SIPI Channel Control Register 3 (SIPI_CCR3)	32	R/W	0000_0000h	51.15.22/ 2236
80	SIPI Channel Status Register 3 (SIPI_CSR3)	32	R	0000_0000h	51.15.23/ 2240

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	32	R/W	0000_0000h	51.15.24/ 2241
8C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	32	R/W	0000_00FFh	51.15.25/ 2242
90	SIPI Channel CRC Register 3 (SIPI_CCRC3)	32	R	0000_0000h	51.15.26/ 2243
94	SIPI Channel Address Register 3 (SIPI_CAR3)	32	R/W	0000_0000h	51.15.27/ 2243
98	SIPI Channel Data Register 3 (SIPI_CDR3)	32	R/W	0000_0000h	51.15.28/ 2244
9C	SIPI Module Configuration Register (SIPI_MCR)	32	R/W	See section	51.15.29/ 2245
A0	SIPI Status Register (SIPI_SR)	32	R	0000_0000h	51.15.30/ 2248
A4	SIPI Max Count Register (SIPI_MAXCR)	32	R/W	FFFF_FFFCh	51.15.31/ 2250
A8	SIPI Address Reload Register (SIPI_ARR)	32	R/W	0000_0000h	51.15.32/ 2250
AC	SIPI Address Count Register (SIPI_ACR)	32	R/W	0000_0000h	51.15.33/ 2251
B0	SIPI Error Register (SIPI_ERR)	32	R	0000_0000h	51.15.34/ 2252

51.15.1 SIPI Channel Control Register 0 (SIPI_CCR0)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR0[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR0[CB] = 0).

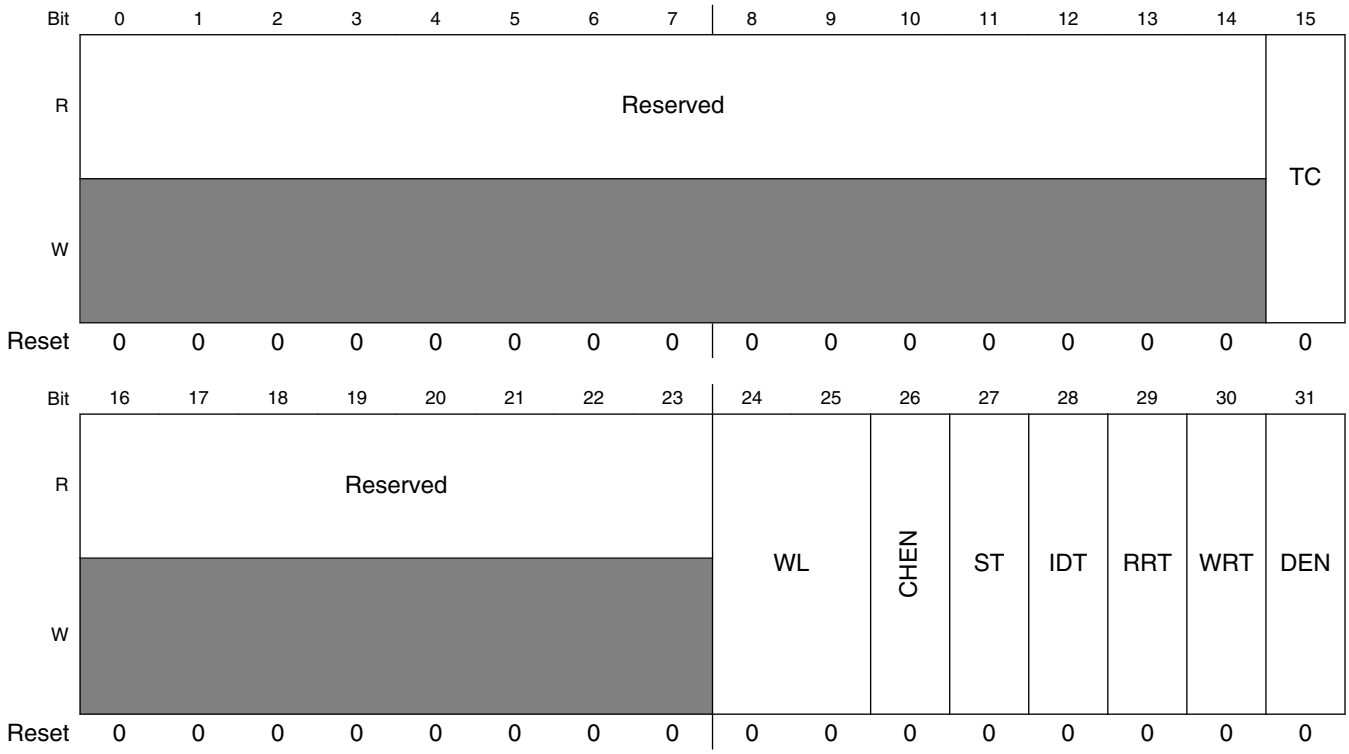
PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 0 (SIPI_CSR0)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 0 (SIPI_CAR0)).

Address: 0h base + 0h offset = 0h



SIPI_CCR0 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR0. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority.

Table continues on the next page...

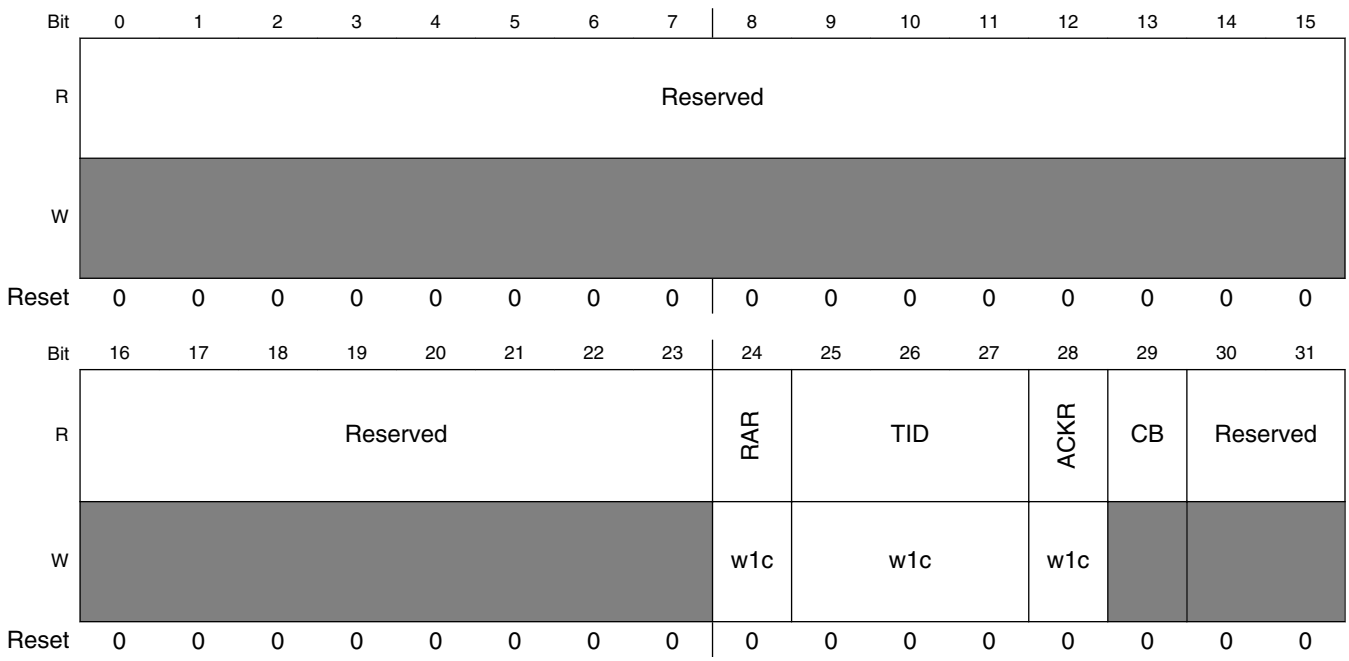
SIPI_CCR0 field descriptions (continued)

Field	Description
	0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1 or SIPI_CCR0[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

51.15.2 SIPI Channel Status Register 0 (SIPI_CSR0)

CSR0 contains the status bits for the current transfer.

Address: 0h base + 4h offset = 4h



SIPI_CSR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 0 status. 0 Channel 0 free 1 Channel 0 busy

Table continues on the next page...

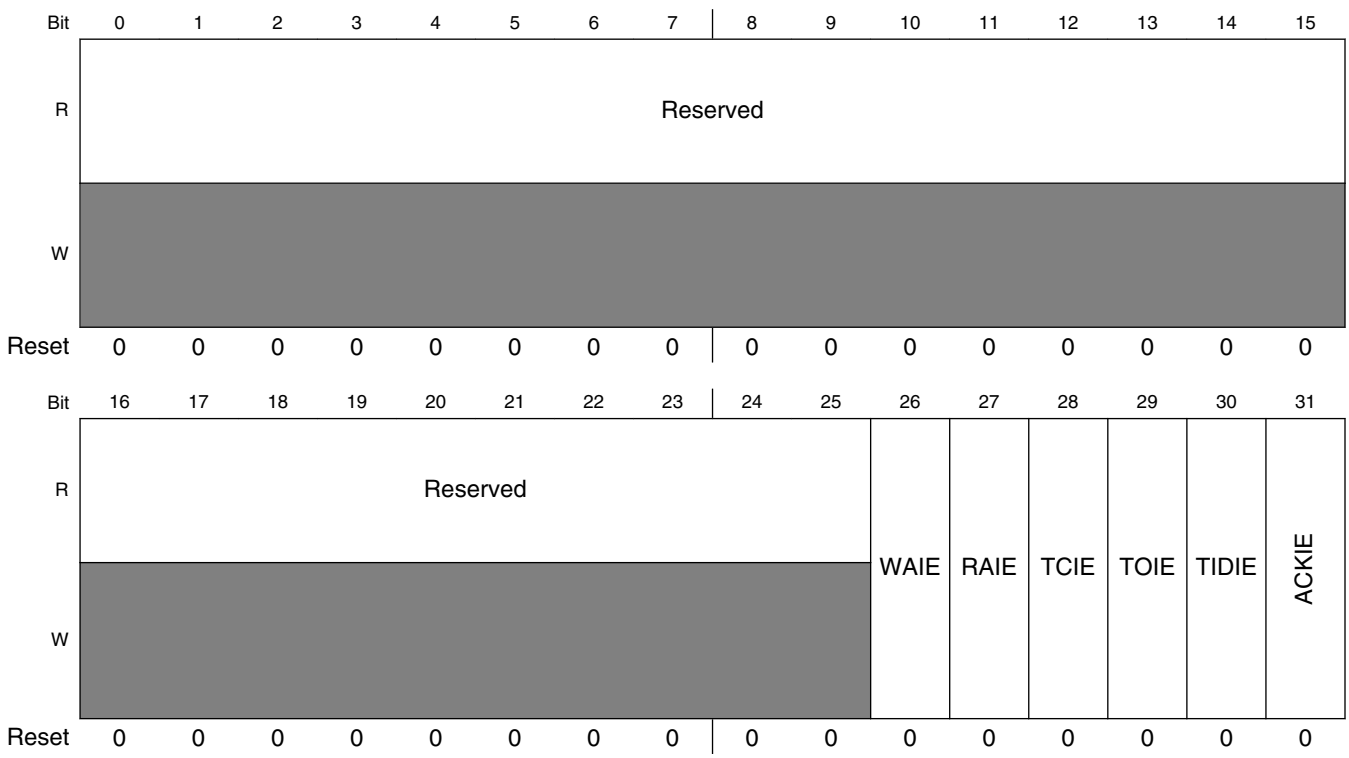
SIPI_CSR0 field descriptions (continued)

Field	Description
30–31 Reserved	This field is reserved.

51.15.3 SIPI Channel Interrupt Register 0 (SIPI_CIR0)

SIPI_CIR0 contains the interrupt enable bits for channel 0.

Address: 0h base + Ch offset = Ch



SIPI_CIR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SIPI_CIR0 field descriptions (continued)

Field	Description
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

51.15.4 SIPI Channel Timeout Register 0 (SIPI_CTOR0)

SIPI_CTOR0 contains the timeout value for Tx requests.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																TOR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

51.15.5 SIPI Channel CRC Register 0 (SIPI_CCRC0)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC0 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

51.15.6 SIPI Channel Address Register 0 (SIPI_CAR0)

SIPI_CAR0 is the address target for data transmission.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR0 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

51.15.7 SIPI Channel Data Register 0 (SIPI_CDR0)

SIPI_CDR0 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR0 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

51.15.8 SIPI Channel Control Register 1 (SIPI_CCR1)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR1[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR1[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 1 (SIPI_CSR1)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 1 (SIPI_CAR1)).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR1 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR1. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR1 field descriptions (continued)

Field	Description
	0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1 or SIPI_CCR1[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

51.15.9 SIPI Channel Status Register 1 (SIPI_CSR1)

CSR1 contains the status bits for the current transfer.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W									w1c	w1c			w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

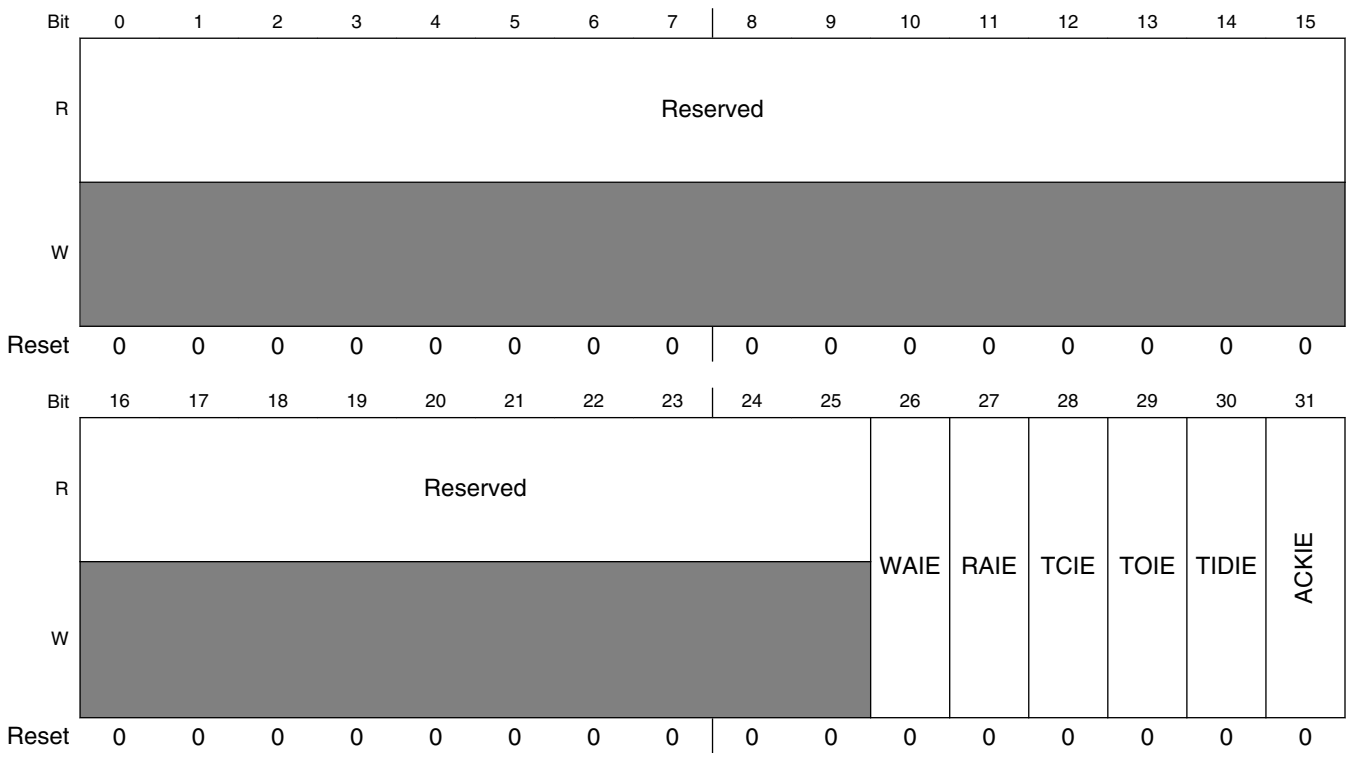
SIPI_CSR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 1 status. 0 Channel 1 free 1 Channel 1 busy
30–31 Reserved	This field is reserved.

51.15.10 SIPI Channel Interrupt Register 1 (SIPI_CIR1)

The CIR1 contains the interrupt enable bits for channel 1.

Address: 0h base + 2Ch offset = 2Ch



SIPI_CIR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR1 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

51.15.11 SIPI Channel Timeout Register 1 (SIPI_CTOR1)

SIPI_CTOR1 contains the timeout value for Tx requests.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																TOR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

51.15.12 SIPI Channel CRC Register 1 (SIPI_CCRC1)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC1 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

51.15.13 SIPI Channel Address Register 1 (SIPI_CAR1)

SIPI_CAR1 is the address target for data transmission.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR1 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

51.15.14 SIPI Channel Data Register 1 (SIPI_CDR1)

SIPI_CDR1 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR1 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

51.15.15 SIPI Channel Control Register 2 (SIPI_CCR2)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR2[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR2[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 2 (SIPI_CSR2)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 2 (SIPI_CAR2)).

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR2 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR2. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
	0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1 or SIPI_CCR2[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

51.15.16 SIPI Channel Status Register 2 (SIPI_CSR2)

SIPI_CSR2 contains the status bits for the current transfer.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W									w1c	w1c			w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

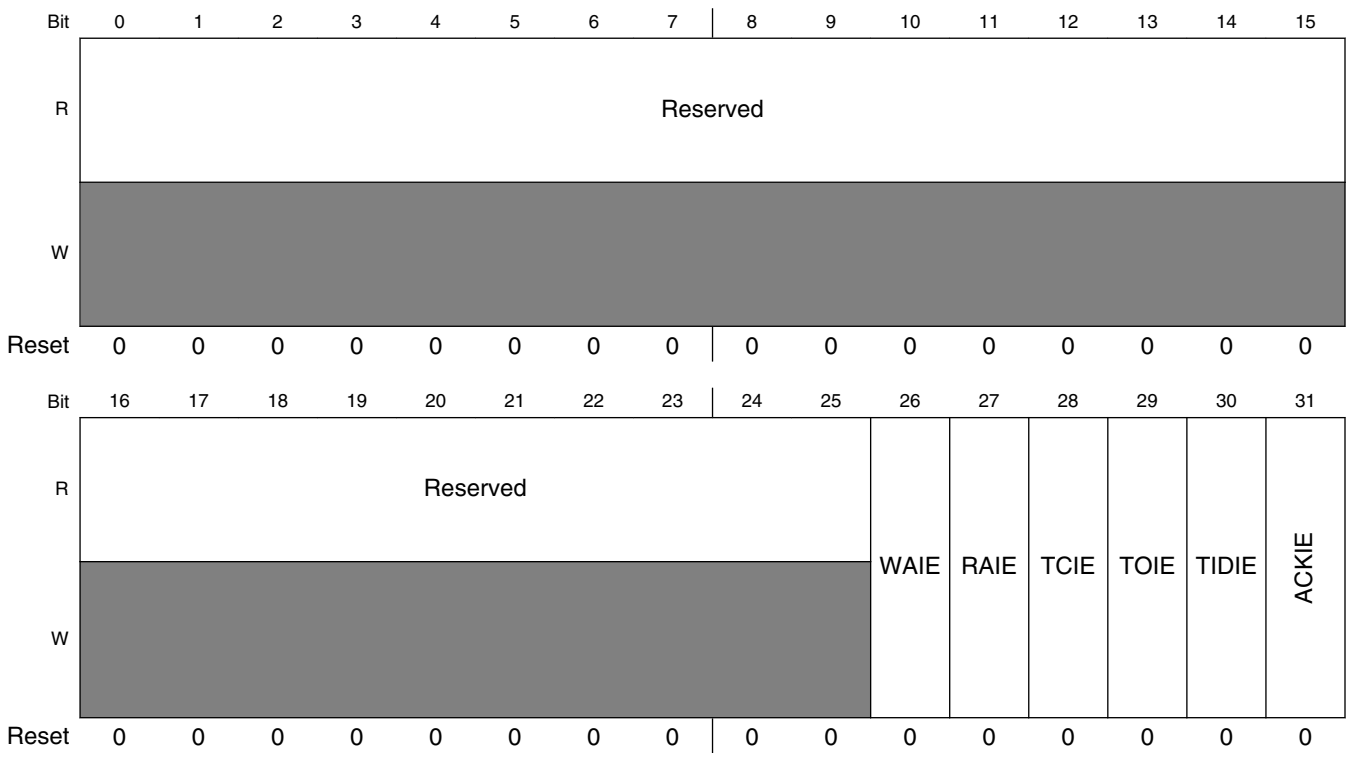
SIPI_CSR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 2 status. 0 Channel 2 free 1 Channel 2 busy
30–31 Reserved	This field is reserved.

51.15.17 SIPI Channel Interrupt Register 2 (SIPI_CIR2)

The SIPI_CIR2 contains the interrupt enable bits for channel 2.

Address: 0h base + 4Ch offset = 4Ch



SIPI_CIR2 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR2 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

51.15.18 SIPI Channel Timeout Register 2 (SIPI_CTOR2)

SIPI_CTOR2 contains the timeout value for Tx requests.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

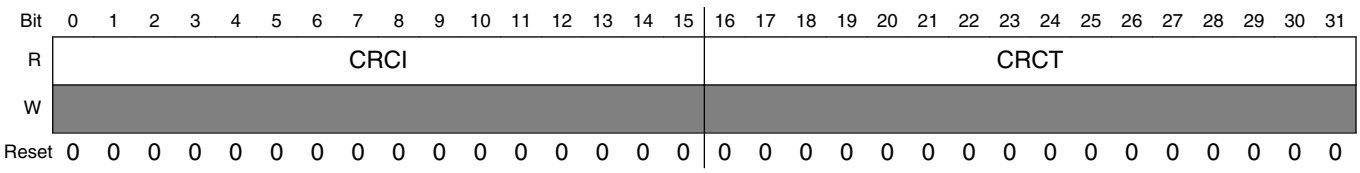
SIPI_CTOR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

51.15.19 SIPI Channel CRC Register 2 (SIPI_CCRC2)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 54h offset = 54h



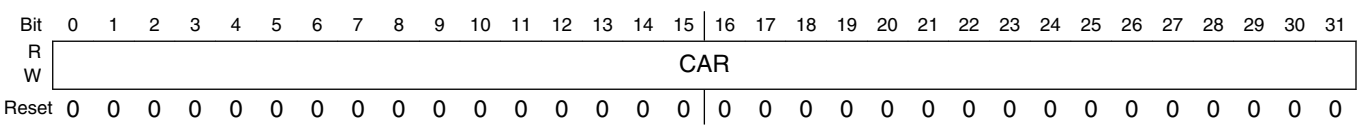
SIPI_CCRC2 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

51.15.20 SIPI Channel Address Register 2 (SIPI_CAR2)

SIPI_CAR2 is the address target for data transmission. For streaming operations this register is the start address.

Address: 0h base + 58h offset = 58h



SIPI_CAR2 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

51.15.21 SIPI Channel Data Register 2 (SIPI_CDR2_*n*)

SIPI_CDR2_*n*contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 5Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR2_*n* field descriptions

Field	Description
0–31 CDR2	Data register bits. Contains the data that will be transmitted, or received.

51.15.22 SIPI Channel Control Register 3 (SIPI_CCR3)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR3[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR3[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 3 (SIPI_CSR3)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 3 (SIPI_CAR3)).

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR3 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR3. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT

Table continues on the next page...

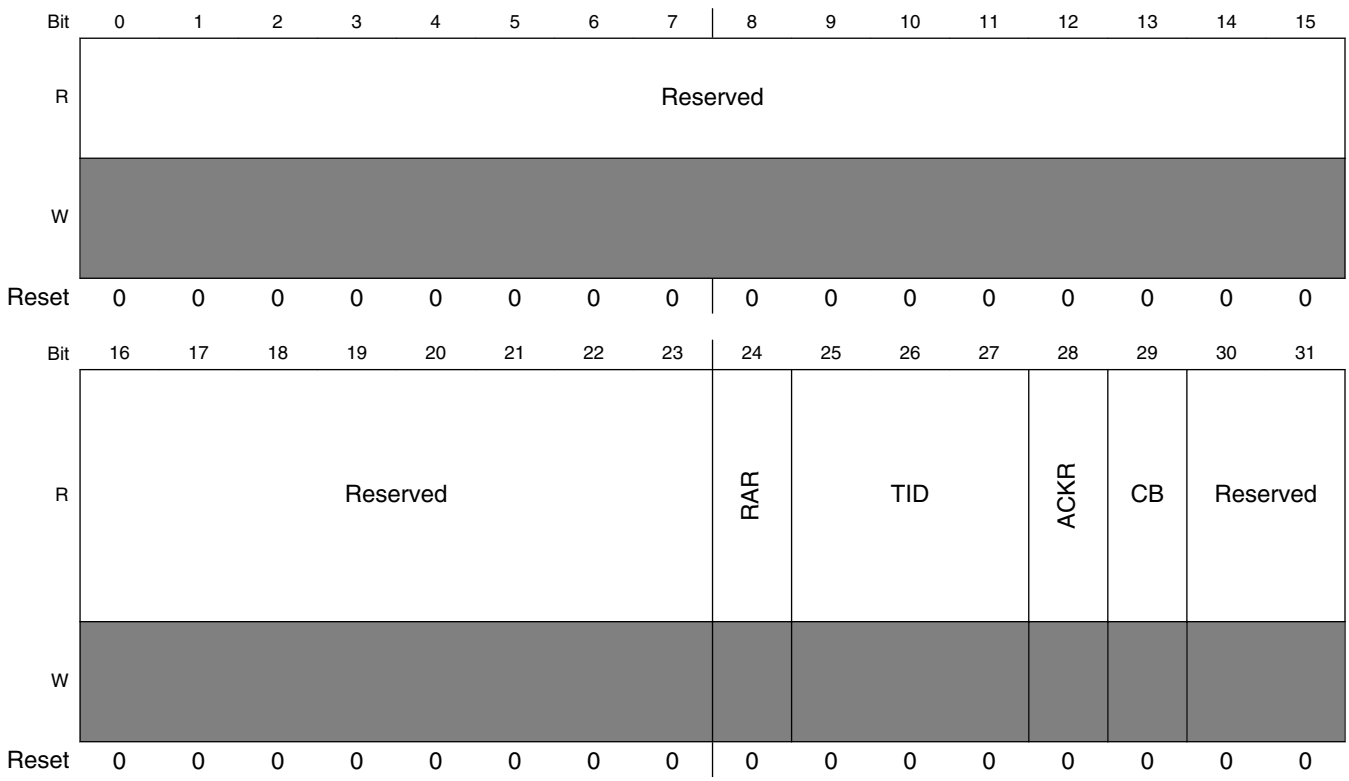
SIPI_CCR3 field descriptions (continued)

Field	Description
	3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1 or SIPI_CCR3[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

51.15.23 SIPI Channel Status Register 3 (SIPI_CSR3)

SIPI_CSR3 contains the status bits for the current transfer.

Address: 0h base + 80h offset = 80h



SIPI_CSR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received

Table continues on the next page...

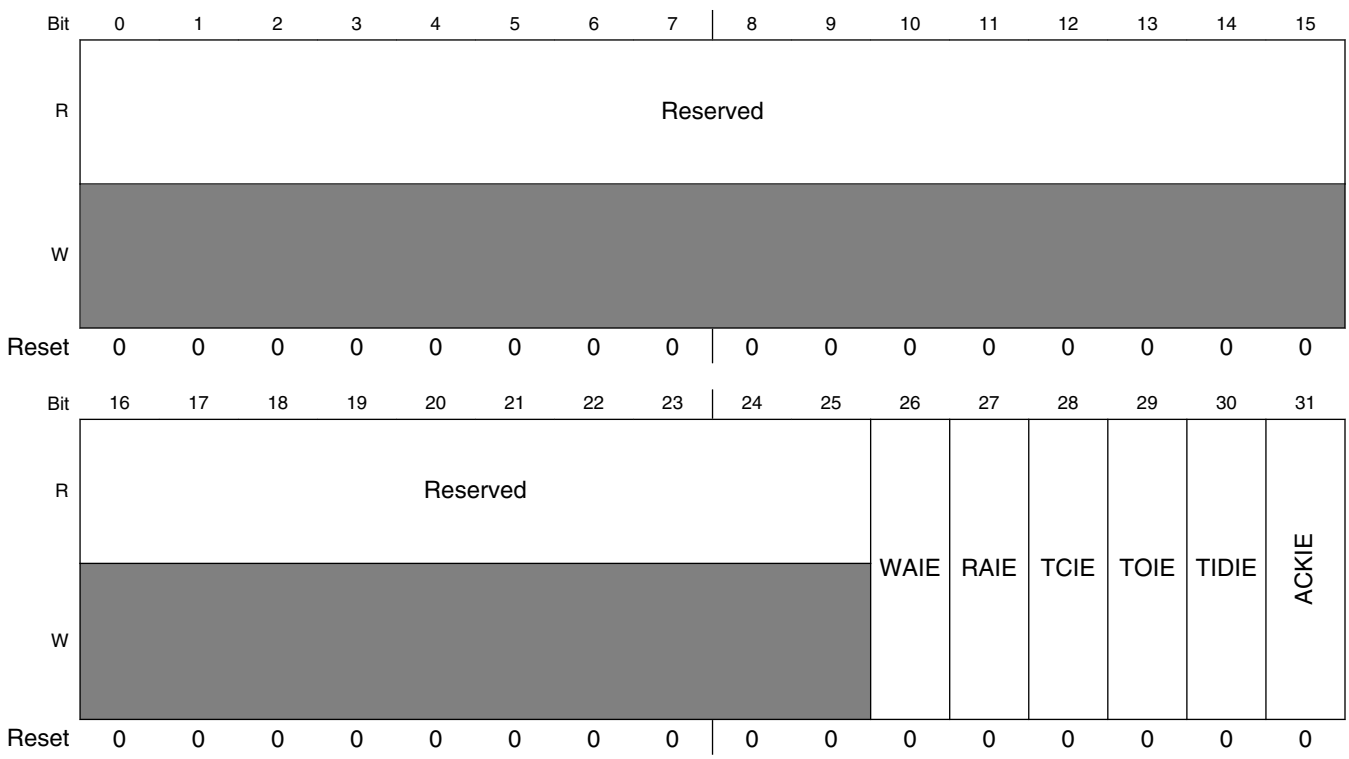
SIPI_CSR3 field descriptions (continued)

Field	Description
29 CB	Channel Busy. Indicates channel 3 status. 0 Channel 3 free 1 Channel 3 busy
30–31 Reserved	This field is reserved.

51.15.24 SIPI Channel Interrupt Register 3 (SIPI_CIR3)

SIPI_CIR3 contains the interrupt enable bits for channel 3.

Address: 0h base + 88h offset = 88h



SIPI_CIR3 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable.

Table continues on the next page...

SIPI_CIR3 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

51.15.25 SIPI Channel Timeout Register 3 (SIPI_CTOR3)

SIPI_CTOR3 contains the timeout value for Tx requests.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																TOR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

51.15.26 SIPI Channel CRC Register 3 (SIPI_CCRC3)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC3 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

51.15.27 SIPI Channel Address Register 3 (SIPI_CAR3)

SIPI_CAR3 is the address target for data transmission.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

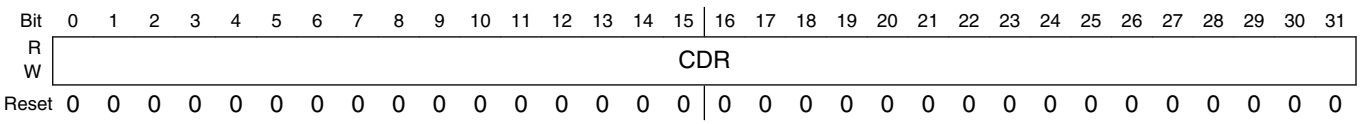
SIPI_CAR3 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

51.15.28 SIPI Channel Data Register 3 (SIPI_CDR3)

SIPI_CDR3 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 98h offset = 98h



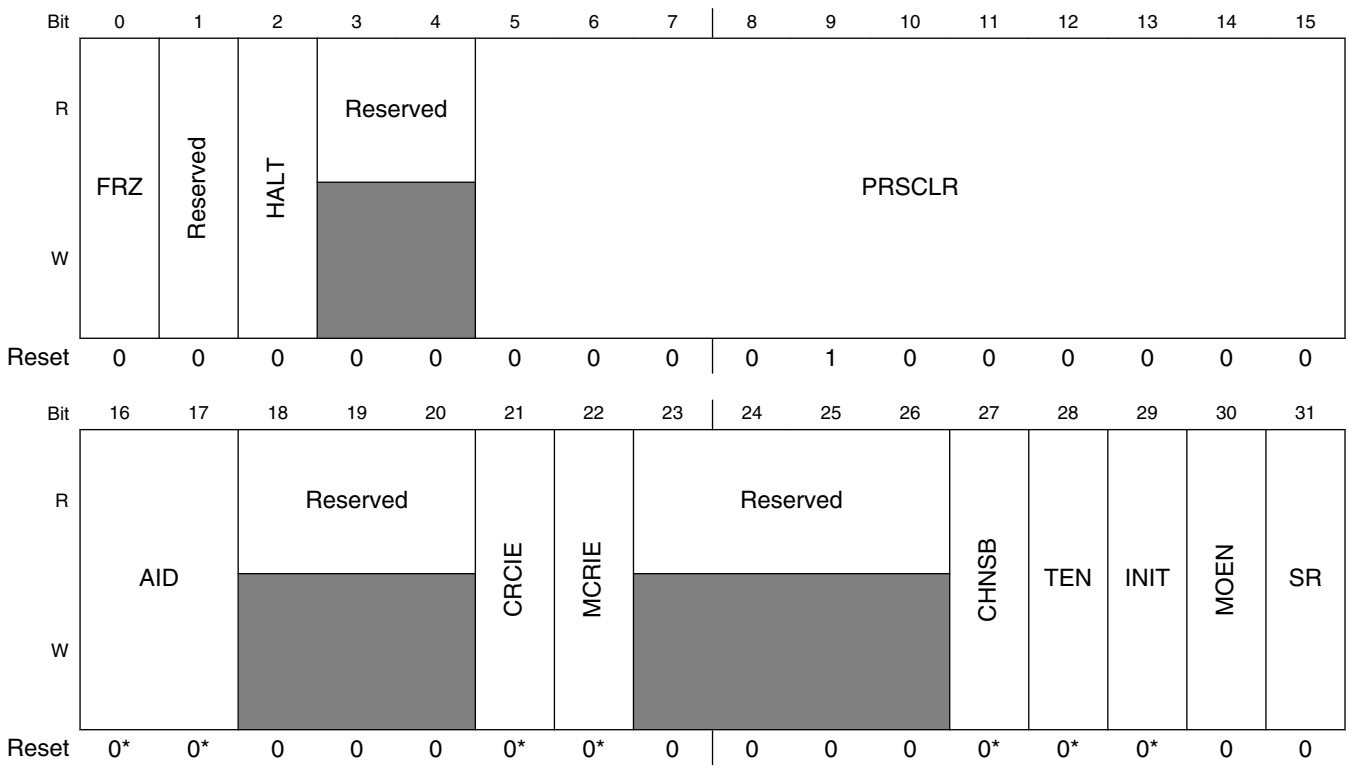
SIPI_CDR3 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

51.15.29 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a global 32-bit configuration register.

Address: 0h base + 9Ch offset = 9Ch



* Notes:
- INIT field: Can only be written when SIPI_MCR[MOEN] = 1.
- TEN field: Can only be written when SIPI_MCR[MOEN] = 1.
- CHNSB field: Can be written only once after reset.
- MCRIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- CRCIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- AID field: Can be written in initialization mode only (SIPI_MCR[INIT] = 1).

SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1.

Table continues on the next page...

SIPI_MCR field descriptions (continued)

Field	Description
	0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1 Reserved	This field is reserved. **Important:** Always write the default value to this field.
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4 Reserved	This field is reserved.
5–15 PRSCLR	Timeout counter prescaler The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are (all other values are ignored): 040h 64 (default) 080h 128 100h 256 200h 512 400h 1024 **NOTE:** This field should be programmed by software during initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. **NOTE:** Writes to SIPI_MCR[PRSCLR] can only be accomplished with 16-bit or 32-bit writes.
16–17 AID	Address Increment/Decrement These bits define the type of address change at the target node. **NOTE:** This bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4 11 not used
18–20 Reserved	This field is reserved.
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable

Table continues on the next page...

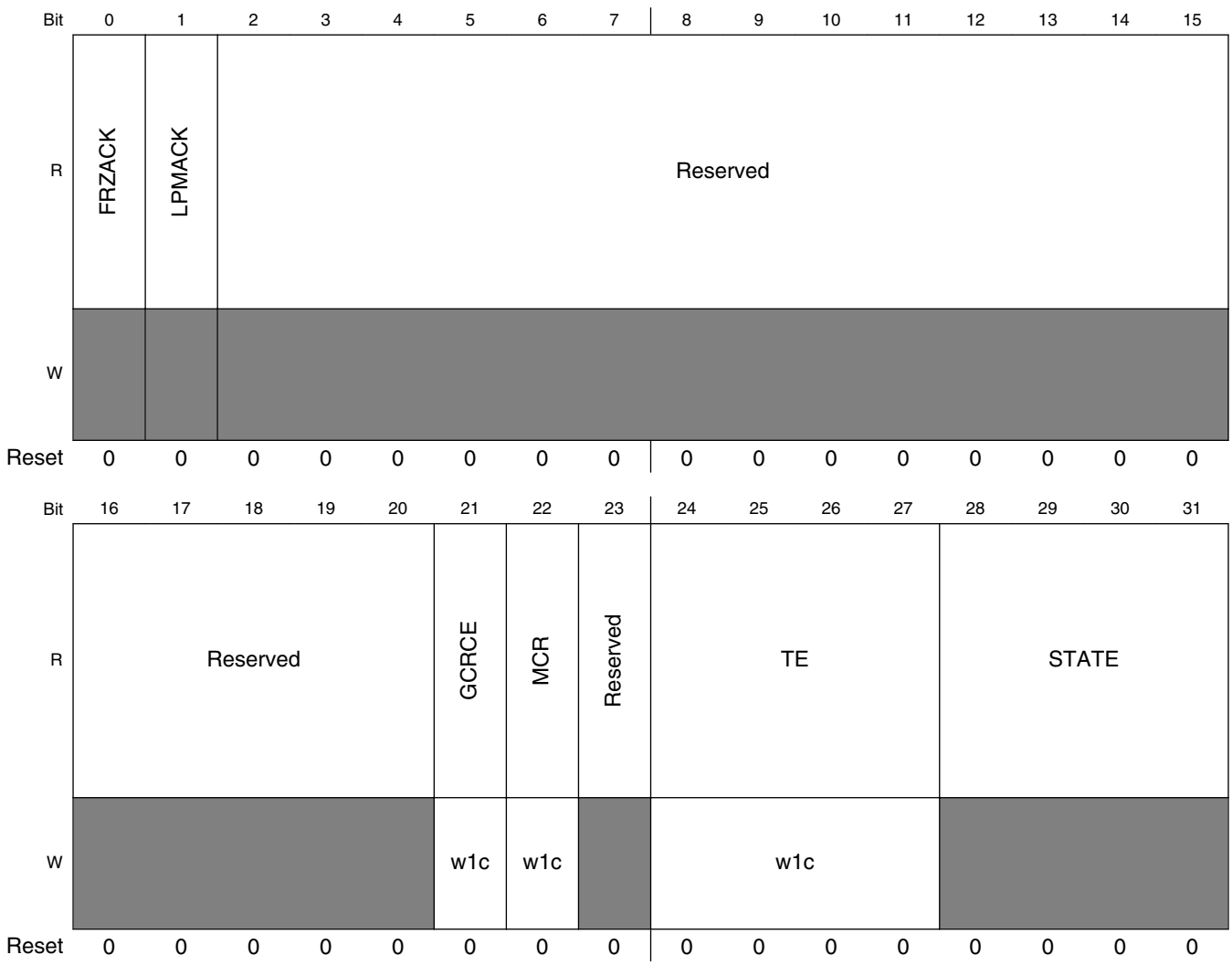
SIPI_MCR field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
23–26 Reserved	This field is reserved.
27 CHNSB	Channel coding select bit. 0 Code Table II (see Table 51-2) 1 Code Table I (see Table 51-2)
28 TEN	Target Enable Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in "streaming without ACK" mode. This bit has to be enabled for the transmission operations also.
29 INIT	Initialization Mode Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits can't be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, all SIPI operations are immediately stopped.
31 SR	Soft Reset Setting this bit clears all status and error registers, and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

51.15.30 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Address: 0h base + A0h offset = A0h



SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited.

Table continues on the next page...

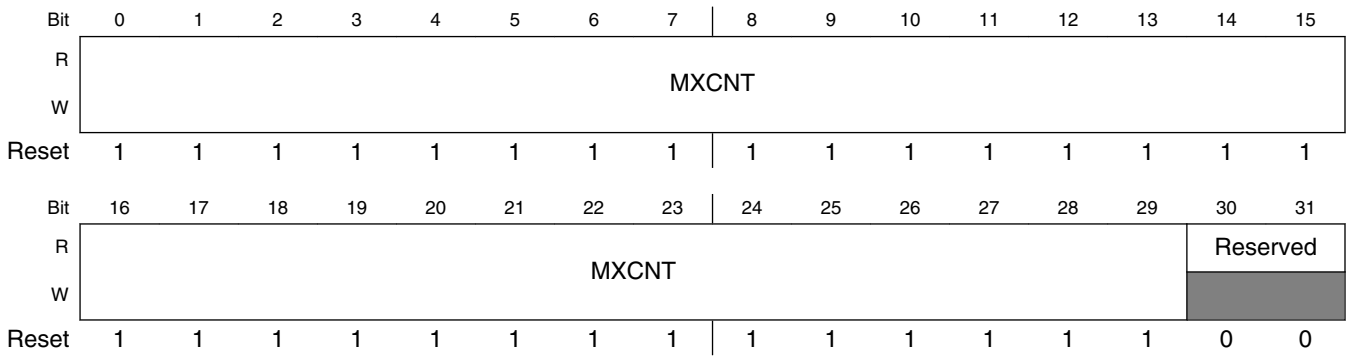
SIPI_SR field descriptions (continued)

Field	Description
	0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge. This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI is not in low power mode 1 SIPI is in Disable Mode.
2–20 Reserved	This field is reserved.
21 GCRCE	Global CRC Error Bit. 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This it should be cleared by software.
23 Reserved	This field is reserved.
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. xxx1 TE0 = 1 - Channel 0 trigger event xx1x TE1 = 1 - Channel 1 trigger event x1xx TE2 = 1 - Channel 2 trigger event 1xxx TE3 = 1 - Channel 3 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

51.15.31 SIPI Max Count Register (SIPI_MAXCR)

SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Address: 0h base + A4h offset = A4h



SIPI_MAXCR field descriptions

Field	Description
0–29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.
30–31 Reserved	This field is reserved.

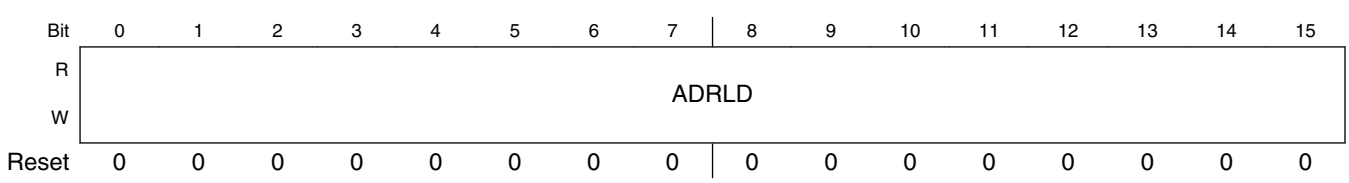
51.15.32 SIPI Address Reload Register (SIPI_ARR)

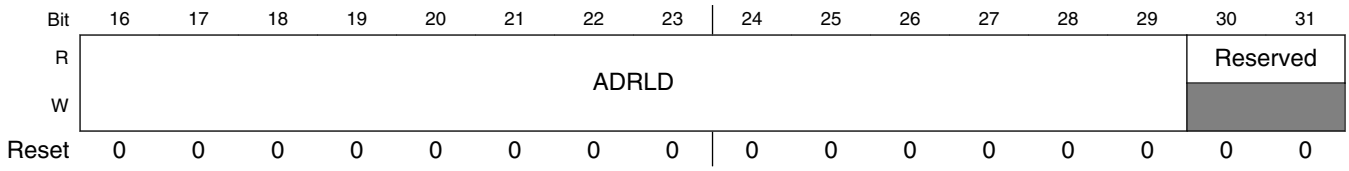
The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.

NOTE

This register is writeable only when SIPI_MCR[INIT] = 1.

Address: 0h base + A8h offset = A8h





SIPI_ARR field descriptions

Field	Description
0–29 ADRLD	ADRLD contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30–31 Reserved	This field is reserved.

51.15.33 SIPI Address Count Register (SIPI_ACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

NOTE

SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation through the initiator.

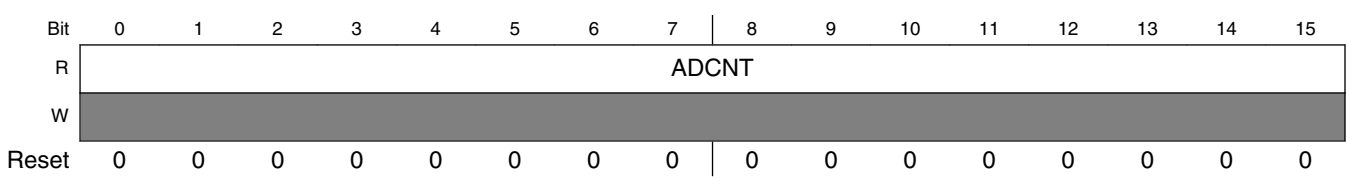
NOTE

When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR (address count register).

NOTE

After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4 or remain the same (depending on configuration). In both cases, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

Address: 0h base + ACh offset = ACh



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADCNT														Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ACR field descriptions

Field	Description
0–29 ADCNT	Feflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will point to the next address to be written.
30–31 Reserved	This field is reserved.

51.15.34 SIPI Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate errors.

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					TOE3	TIDE3	ACKE3	Reserved					TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					TOE1	TIDE1	ACKE1	Reserved					TOE0	TIDE0	ACKE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ERR field descriptions

Field	Description
0–4 Reserved	This field is reserved.
5 TOE3	Timeout Error for Channel 3. 0 Timeout error occured 1 Timeout error didn't occur
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8–12 Reserved	This field is reserved.
13 TOE2	Timeout Error for Channel 2. 0 Timeout error occured 1 Timeout error didn't occur
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16–20 Reserved	This field is reserved.
21 TOE1	Timeout Error for Channel 1. 0 Timeout error occured 1 Timeout error didn't occur
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28 Reserved	This field is reserved.
29 TOE0	Timeout Error for Channel 0. 0 Timeout error occured 1 Timeout error did not occur

Table continues on the next page...

SIPI_ERR field descriptions (continued)

Field	Description
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

Chapter 52
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

52.1 Introduction

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

52.2 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

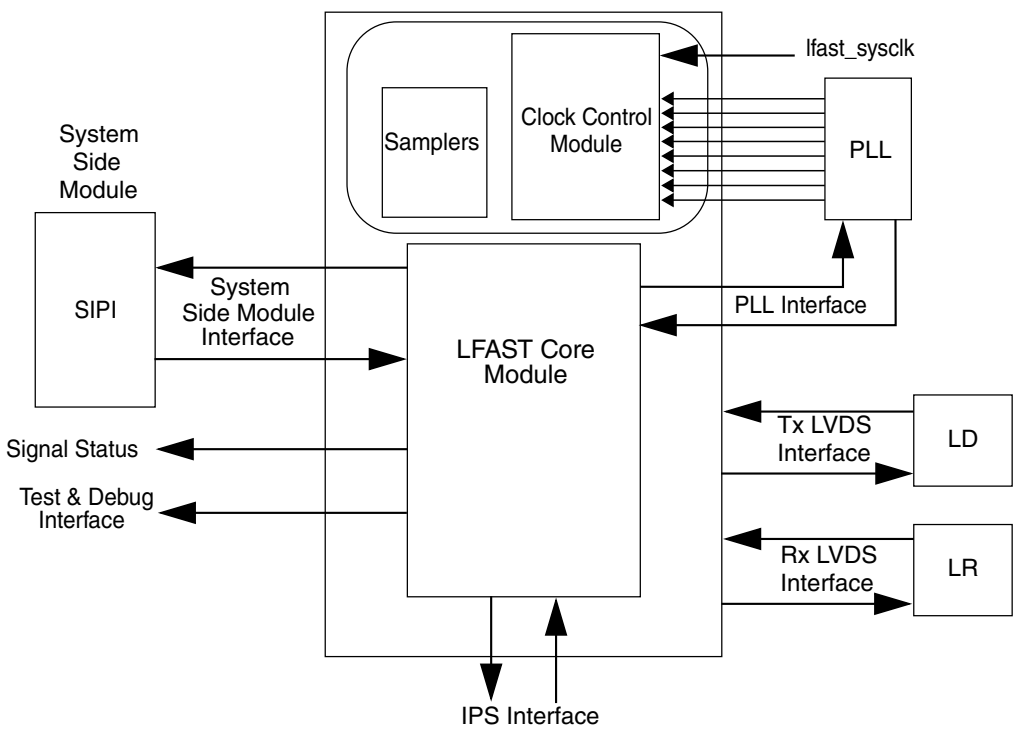


Figure 52-1. LFAST block diagram

52.3 External signals

LFAST is a five pin interface with the following signals :

- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

52.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

52.4 LFAST frame structure

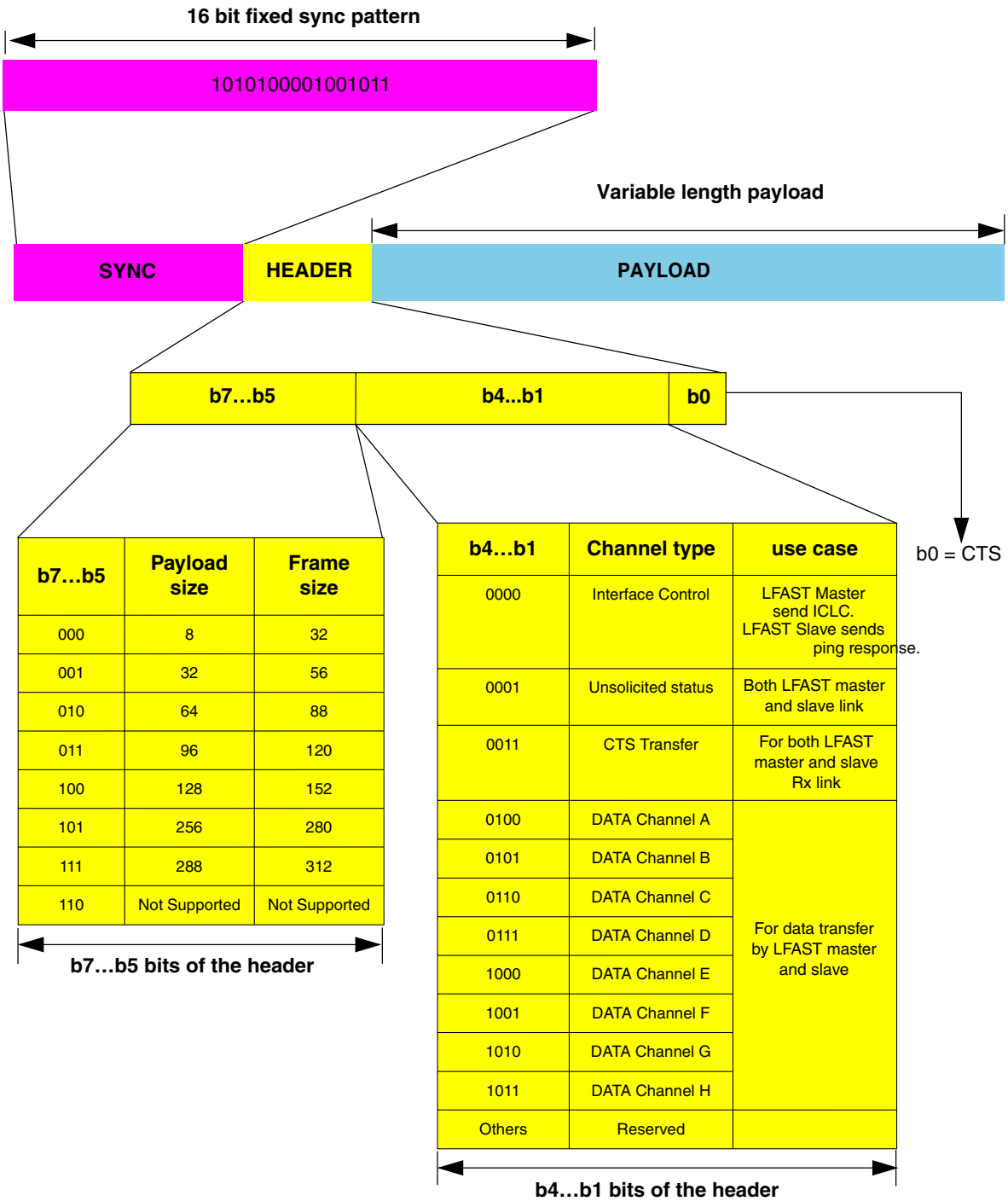
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See Figure 52-2 for details of the LFAST frame.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	288	312
110	Not Supported	Not Supported

b4…b1	Channel type	use case
0000	Interface Control	LFAST Master send ICLC. LFAST Slave sends ping response.
0001	Unsolicited status	Both LFAST master and slave link
0011	CTS Transfer	For both LFAST master and slave Rx link
0100	DATA Channel A	For data transfer by LFAST master and slave
0101	DATA Channel B	
0110	DATA Channel C	
0111	DATA Channel D	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 52-2. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

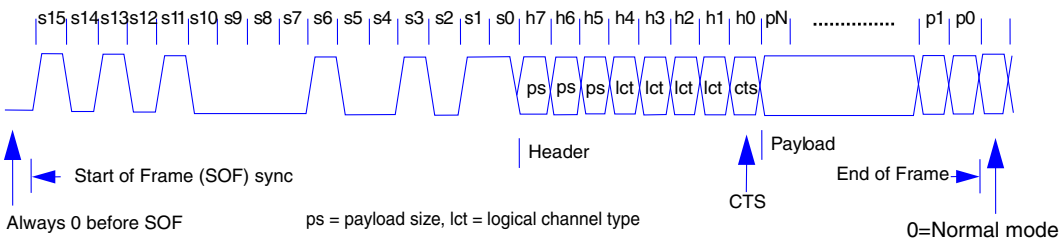


Figure 52-3. Serial frame structure

The frame structure is shown in Figure 52-3 and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in Table 52-1 :

Table 52-1. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.

- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

52.5 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer up to the maximum data rate shown in the product Data Sheet.
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Receives ICLC frames
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

52.6 Memory map and register definition

All registers are 32 bits wide.

LFAST memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	52.6.1/2262
4	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	52.6.2/2264
8	LFAST Correlator Control Register (LFAST_COCR)	32	R/W	0000_000Eh	52.6.3/2265
C	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	52.6.4/2267
10	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	52.6.5/2268
14	LFAST Rate Change Delay Control Register (LFAST_RCDCR)	32	R/W	000F_0000h	52.6.6/2269
18	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	52.6.7/2269
1C	LFAST ICLC Control Register (LFAST_ICR)	32	R/W	0000_0000h	52.6.8/2271
20	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	52.6.9/2272
2C	LFAST Rx FIFO CTS Control Register (LFAST_RFCR)	32	R/W	000F_0009h	52.6.10/ 2272
30	LFAST Tx Interrupt Enable Register (LFAST_TIER)	32	R/W	0000_0000h	52.6.11/ 2273
34	LFAST Rx Interrupt Enable Register (LFAST_RIER)	32	R/W	0000_0000h	52.6.12/ 2274
38	LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)	32	R/W	0000_0000h	52.6.13/ 2276
3C	LFAST PLL Control Register (LFAST_PLLCR)	32	R/W	0000_005Ch	52.6.14/ 2278
40	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	52.6.15/ 2280
44	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	52.6.16/ 2283
48	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	52.6.17/ 2283
4C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	52.6.17/ 2283
50	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	52.6.17/ 2283
54	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	52.6.17/ 2283
58	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	52.6.17/ 2283
5C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	52.6.17/ 2283

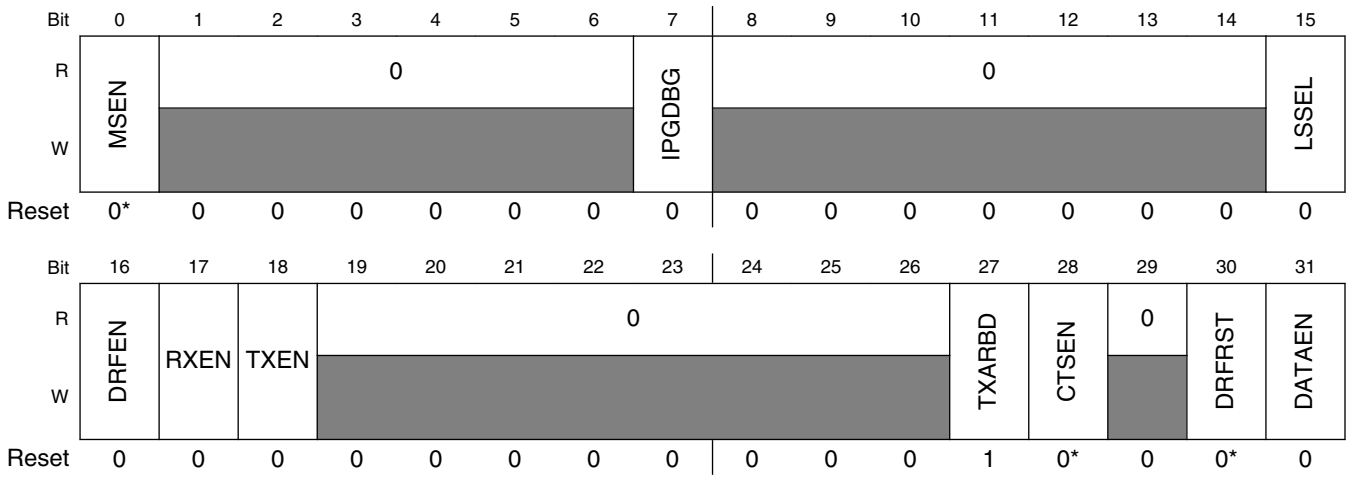
Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	52.6.17/ 2283
64	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	52.6.17/ 2283
68	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	52.6.17/ 2283
80	LFAST Global Status Register (LFAST_GSR)	32	R	See section	52.6.18/ 2284
84	LFAST Ping Status Register (LFAST_PISR)	32	R	0000_0000h	52.6.19/ 2285
94	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	52.6.20/ 2286
98	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	52.6.21/ 2287
9C	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	52.6.22/ 2288
A0	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	R/W	0000_0000h	52.6.23/ 2290
A4	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	52.6.24/ 2292
A8	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	52.6.25/ 2293
AC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	52.6.26/ 2294
B0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	52.6.26/ 2294
B4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	52.6.26/ 2294
B8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	52.6.26/ 2294
BC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	52.6.26/ 2294
C0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	52.6.26/ 2294
C4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	52.6.26/ 2294
C8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	52.6.26/ 2294
CC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	52.6.26/ 2294

52.6.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 0h base + 0h offset = 0h



* Notes:
- DRFRST field: Set by user software and then cleared by system hardware.
- CTSEN field: Only writable when MCR[DRFEN] = 0.
- MSEN field: Writable only once after the asynchronous reset.

LFAST_MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable

Table continues on the next page...

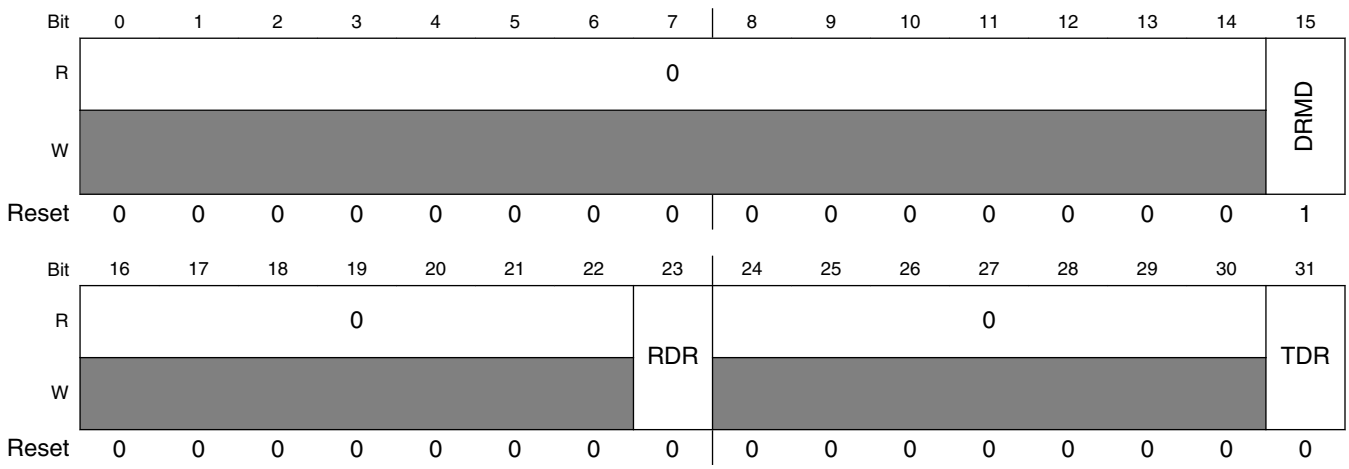
LFAST_MCR field descriptions (continued)

Field	Description
	registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be modified by LFAST slave H/W on reception of an ICLC command frame. This field can only be written in dual mode (LFAST_GSR[DUALMD] = 1). 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 LFAST transmitter Interface is enabled. New requests are accepted.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. This bit is only writable when MCR[DRFEN] = 0. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit doesn't represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset.. 0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

52.6.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 0h base + 4h offset = 4h



LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0 S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no affect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is high speed.

Table continues on the next page...

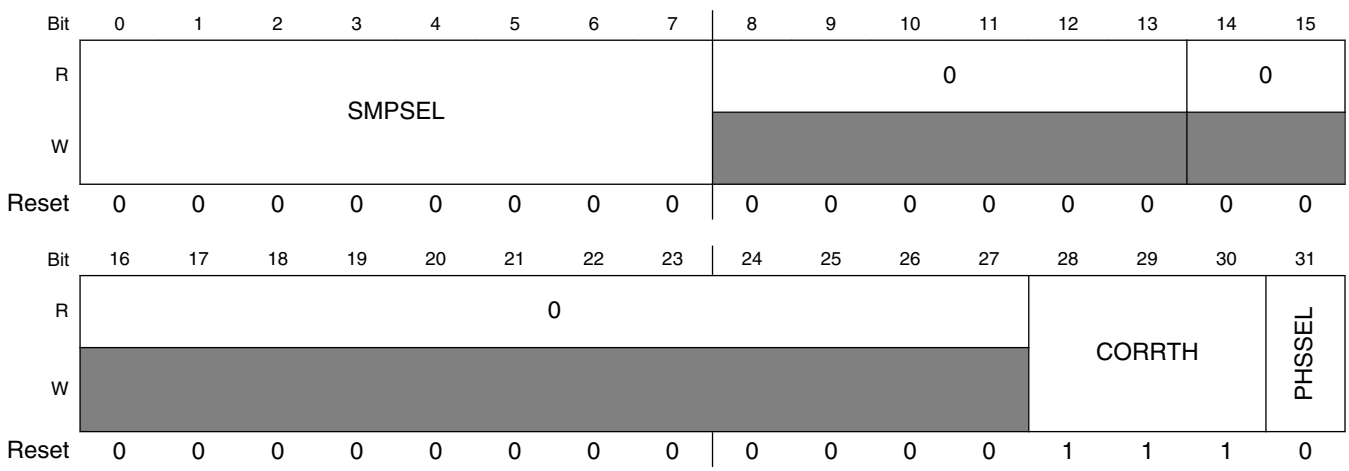
LFAST_SCR field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Tx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is high speed.

52.6.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 0h base + 8h offset = 8h



LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0.

Table continues on the next page...

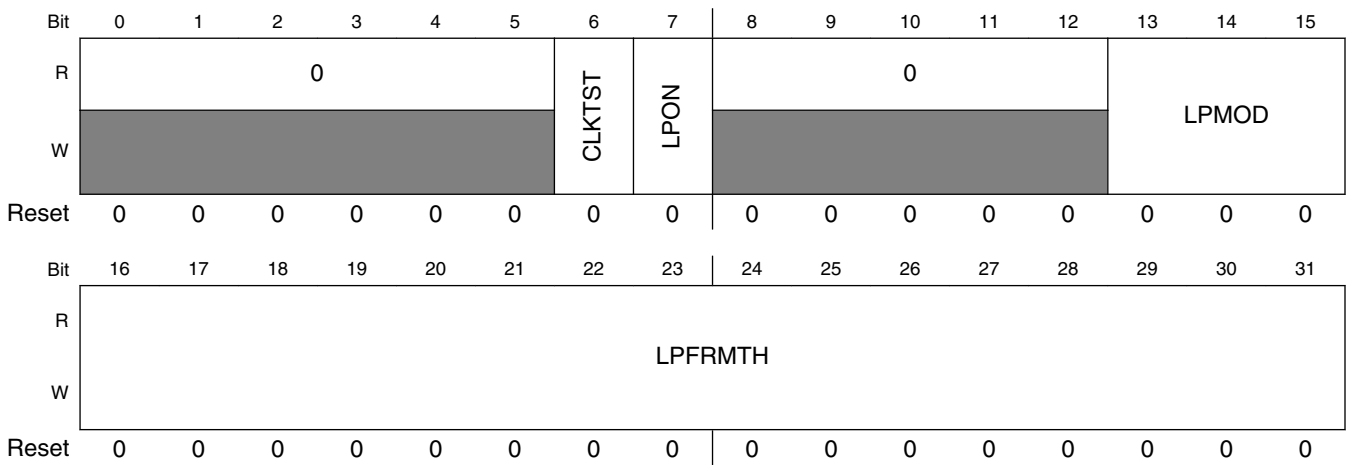
LFAST_COCR field descriptions (continued)

Field	Description
	00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

52.6.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 0h base + Ch offset = Ch



LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved

Table continues on the next page...

LFAST_TMCR field descriptions (continued)

Field	Description
	110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

52.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															LPCNTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload

Table continues on the next page...

LFAST_ALCR field descriptions (continued)

Field	Description
	... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

52.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DRCNT				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT			
W																																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ...

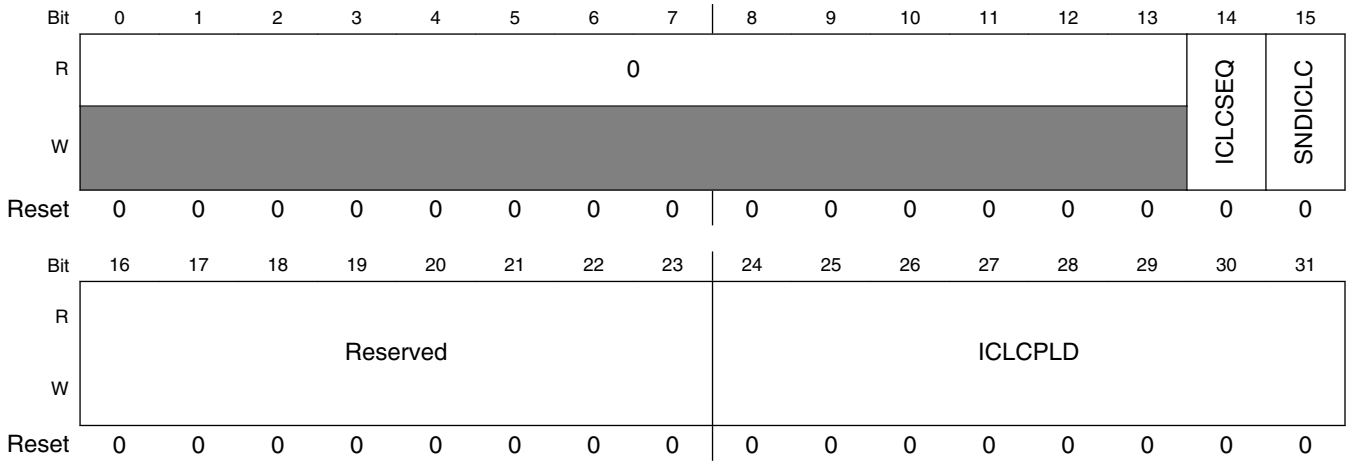
Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
	12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 µs) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 µs) ... Eh 14 cycles Fh 15 cycles

52.6.8 LFAST ICLC Control Register (LFAST_ICR)

Address: 0h base + 1Ch offset = 1Ch

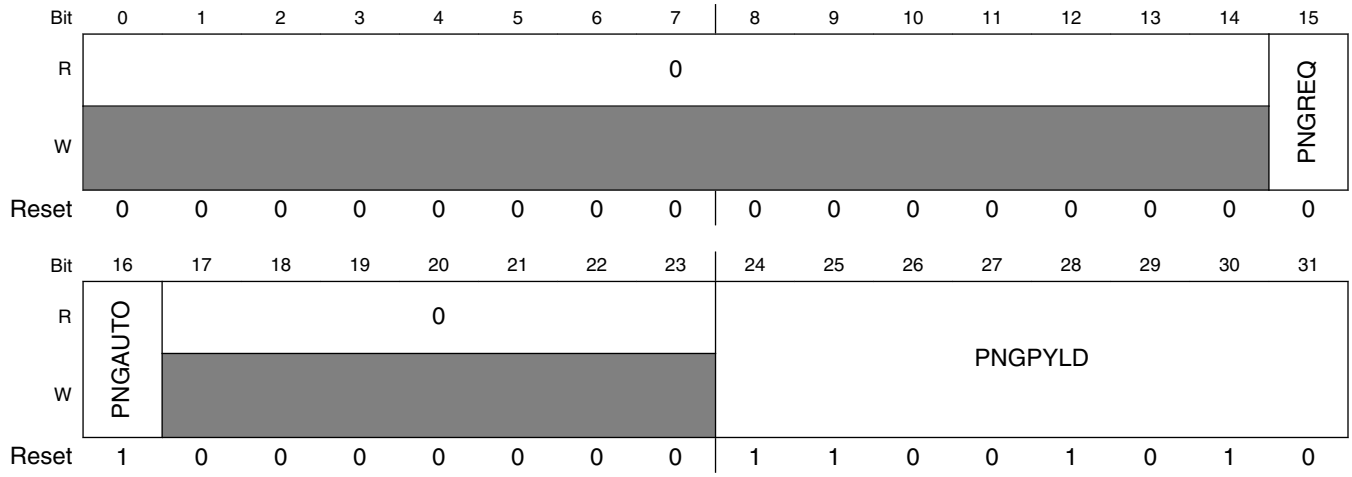


LFAST_ICR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 0 Single ICLC frame transfer. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0.
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. Set by user software and cleared by system hardware. 0 No Valid ICLC frame for transfer. 1 Valid ICLC frame for transfer.
16–23 Reserved	This field is reserved.
24–31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. This field can only be written when ICR[SNDICLC] = 0 (see Table 52-8 for supported ICLC payloads).

52.6.9 LFAST Ping Control Register (LFAST_PICR)

Address: 0h base + 20h offset = 20h

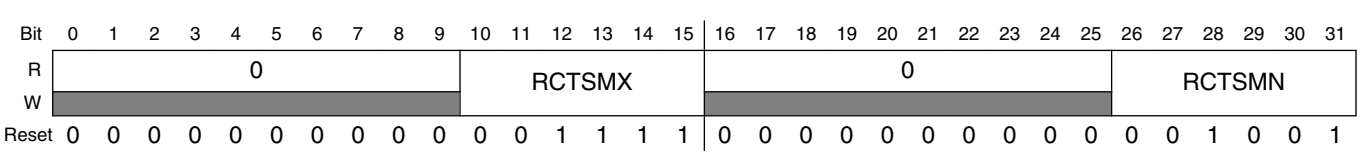


LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

52.6.10 LFAST Rx FIFO CTS Control Register (LFAST_RFCR)

Address: 0h base + 2Ch offset = 2Ch

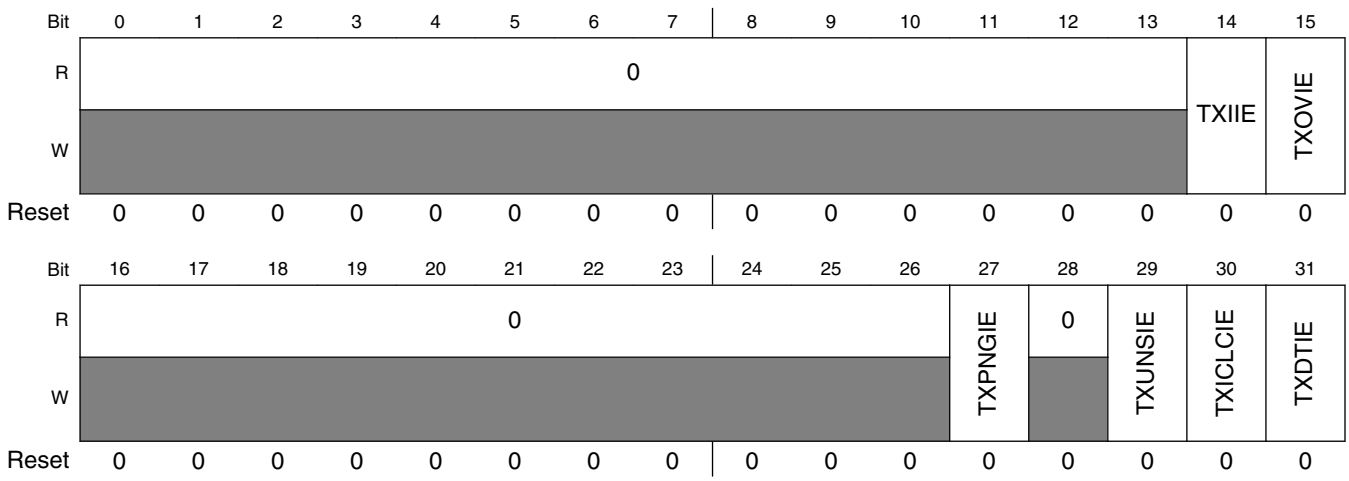


LFAST_RFCR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated. This field can only be written when LFAST_MCR[DRFEN] = 0.
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set. This field can only be written when LFAST_MCR[DRFEN] = 0.

52.6.11 LFAST Tx Interrupt Enable Register (LFAST_TIER)

Address: 0h base + 30h offset = 30h



LFAST_TIER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIIE	Tx Data Interface Not Enabled - (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable.

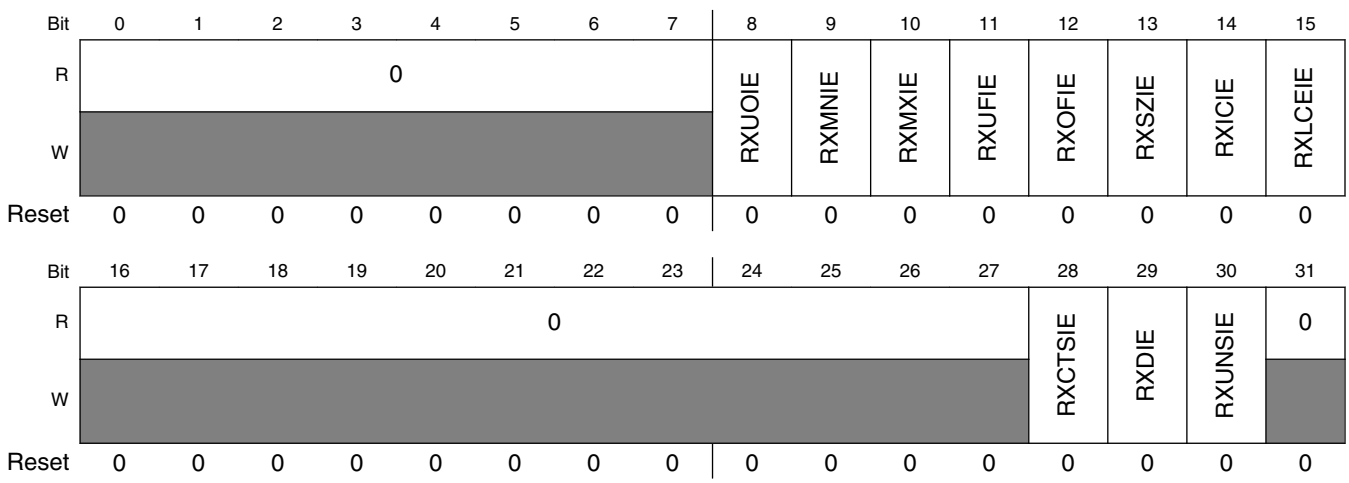
Table continues on the next page...

LFAST_TIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

52.6.12 LFAST Rx Interrupt Enable Register (LFAST_RIER)

Address: 0h base + 34h offset = 34h



LFAST_RIER field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled

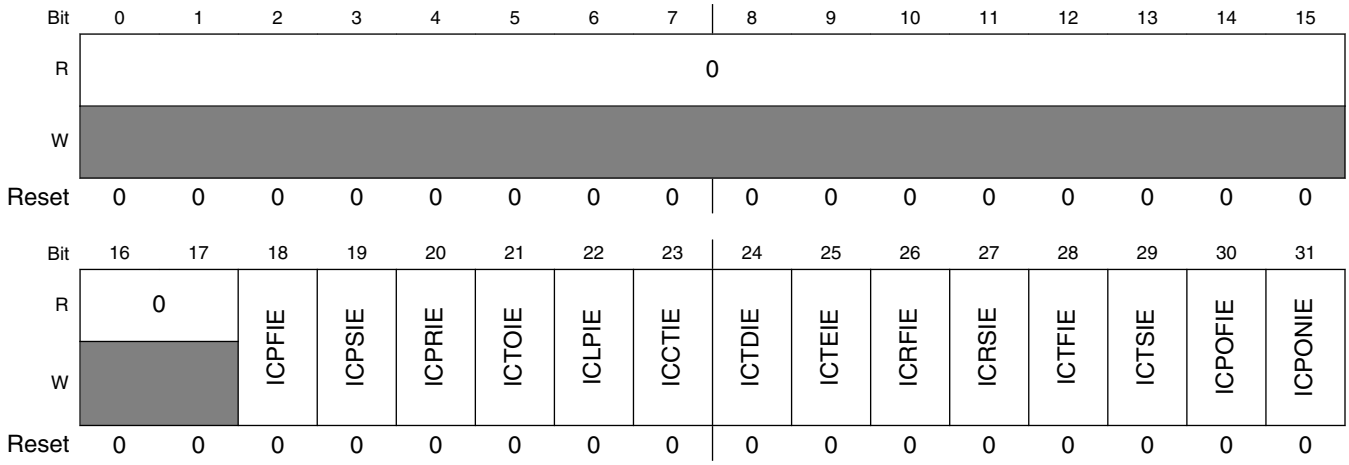
Table continues on the next page...

LFAST_RIER field descriptions (continued)

Field	Description
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in Table 52-12 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.6.13 LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)

Address: 0h base + 38h offset = 38h



LFAST_RIIER field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received

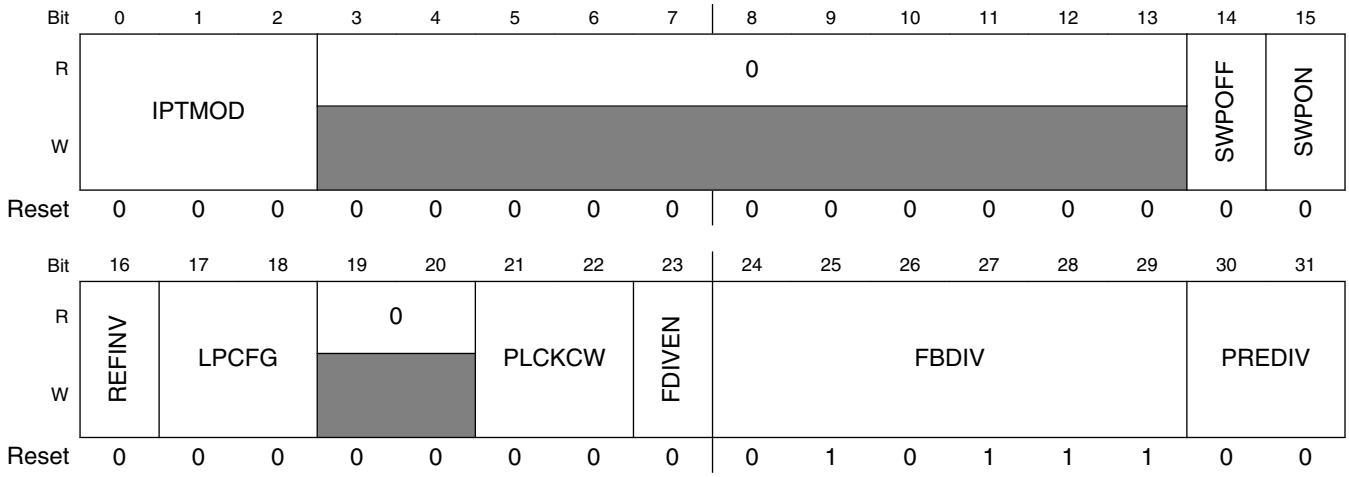
Table continues on the next page...

LFAST_RIIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

52.6.14 LFAST PLL Control Register (LFAST_PLLCR)

Address: 0h base + 3Ch offset = 3Ch



LFAST_PLLCR field descriptions

Field	Description
0–2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
3–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SWPOFF	SW signal to turn OFF the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned OFF.
15 SWPON	SW signal to turn ON the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned ON
16 REFINV	Inverts reference clock edge to PFD. If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Do not inverted 1 Invert

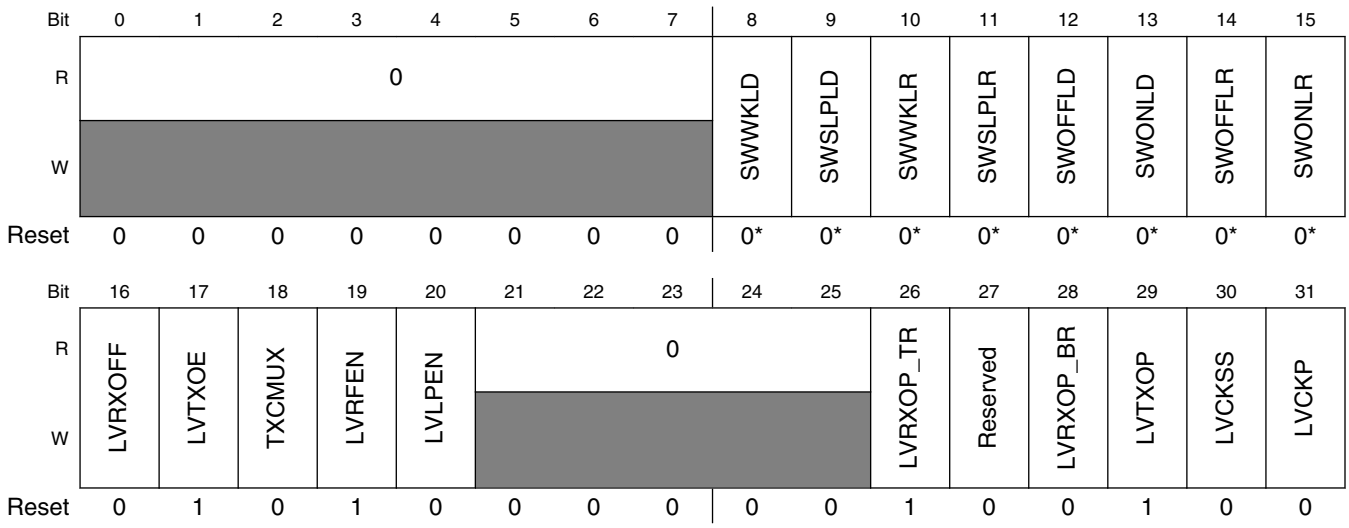
Table continues on the next page...

LFAST_PLLCR field descriptions (continued)

Field	Description
17–18 LPCFG	PLL Loop Optimization for sysclk frequency 00 1 × IBASE current 01 0.5 × IBASE current 10 1.5 × IBASE current 11 2 × IBASE current
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24–29 FBDIV	Feedback Division factor for VCO output clock. This field can only be written when LFAST_MCR[DRFEN] = 0. 00h No clock output 01h – 0Ah Reserved 0Bh Divide by 12 (11.5, if FDIVEN = 1) ... ... 1Fh Divide by 32 (31.5, if FDIVEN = 1) 20h – 3Fh Reserved
30–31 PREDIV	Division factor for PLL Reference Clock input. This field can only be written when LFAST_MCR[DRFEN] = 0. 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

52.6.15 LFAST LVDS Control Register (LFAST_LCR)

Address: 0h base + 40h offset = 40h



* Notes:
- SWONLR field: Set by user software and cleared by system hardware.
- SWOFFLR field: Set by user software and cleared by system hardware.
- SWONLD field: Set by user software and cleared by system hardware.
- SWOFFLD field: Set by user software and cleared by system hardware.
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned OFF(provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)
15 SWONLR	SW signal to turn ON the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled

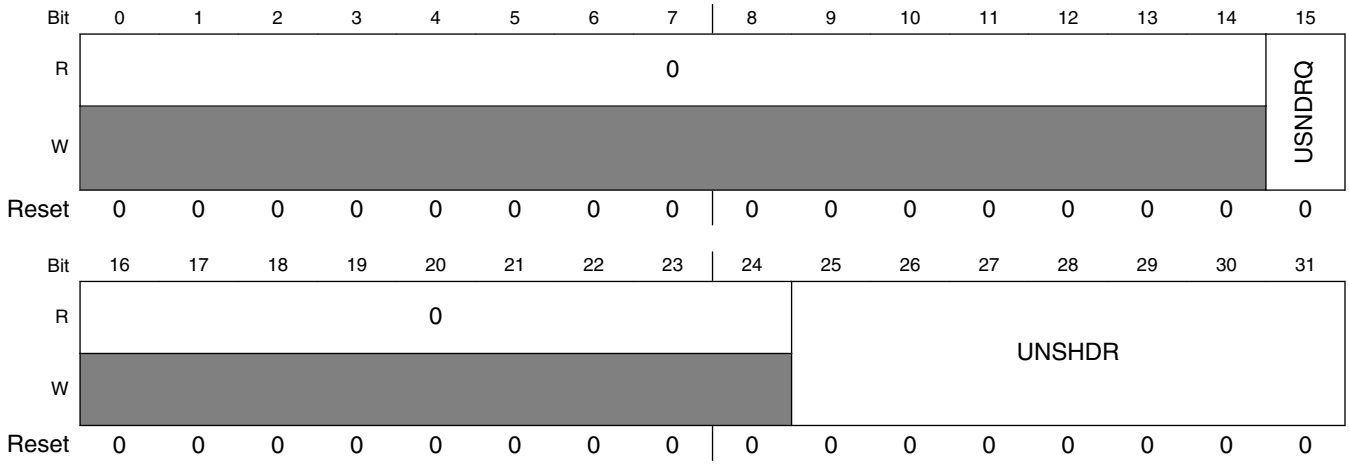
Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
20 LVLPEN	Tx LVDS internal loopback enable The bit is reflected by output signal ipp_digrf_lvds_lpbk_en.The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
27 Reserved	This field is reserved.
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than the maximum baud rate. 1 Required for LFAST receiver maximum baud rate.
29 LVTXOP	Control signal for LFASTand Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 normal clock used to sample the LVDS data 1 adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

52.6.16 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 0h base + 44h offset = 44h

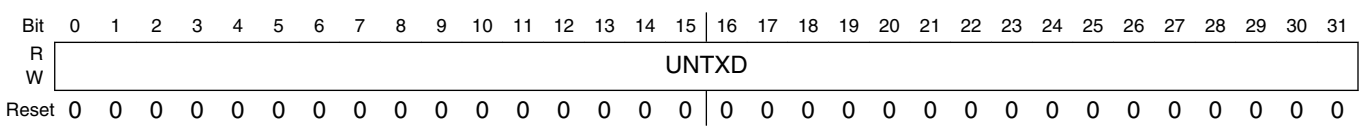


LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

52.6.17 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)

Address: 0h base + 48h offset + (4d × i), where i=0d to 8d

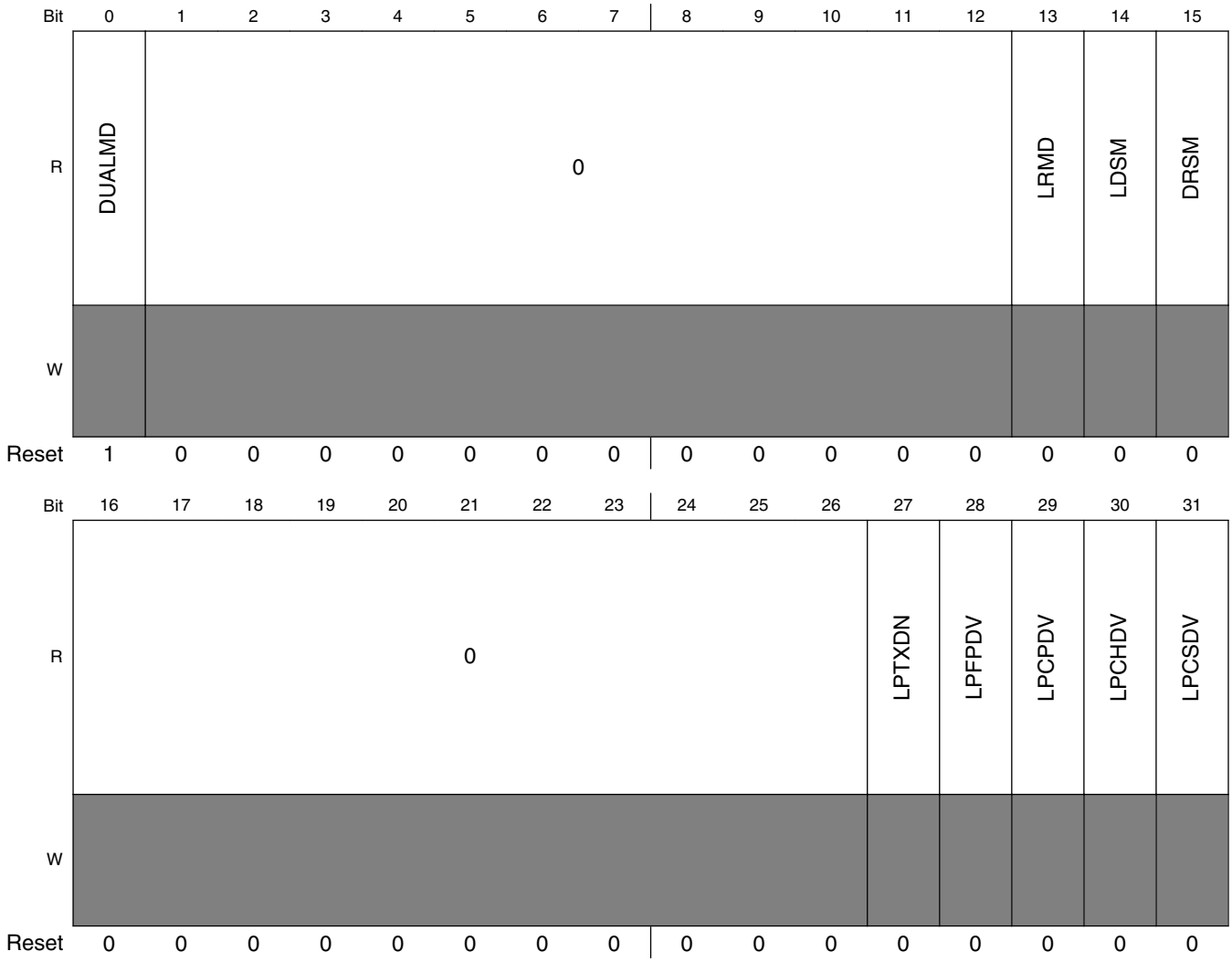


LFAST_UNSTDR*n* field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

52.6.18 LFAST Global Status Register (LFAST_GSR)

Address: 0h base + 80h offset = 80h



LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active.

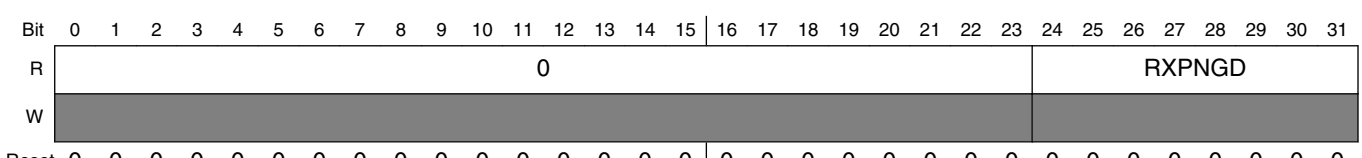
Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

52.6.19 LFAST Ping Status Register (LFAST_PISR)

Address: 0h base + 84h offset = 84h



LFAST_PISR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

52.6.20 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 0h base + 94h offset = 94h

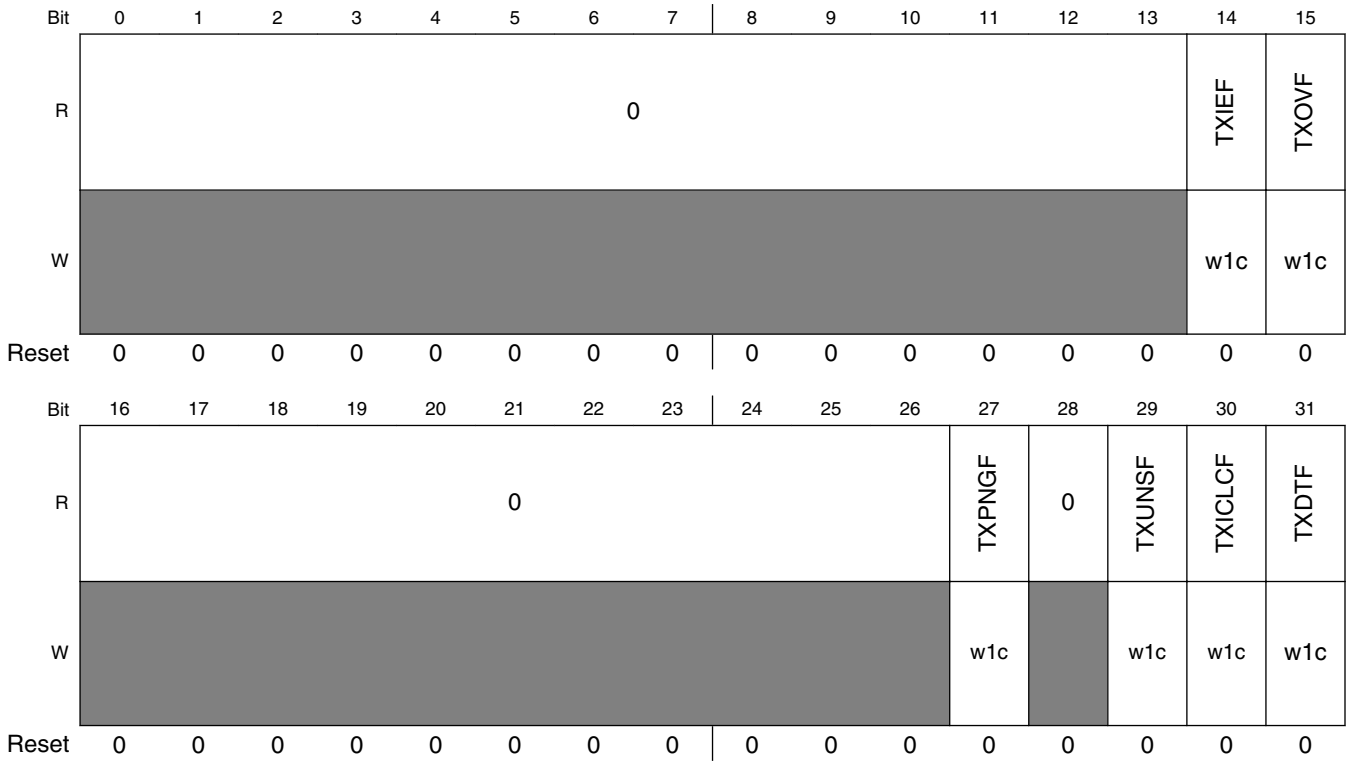
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		RXDCNT						0					RXFCNT			0		TXDCNT						0					TXFCNT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

52.6.21 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 0h base + 98h offset = 98h



LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_TISR field descriptions (continued)

Field	Description
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

52.6.22 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.6.23 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred

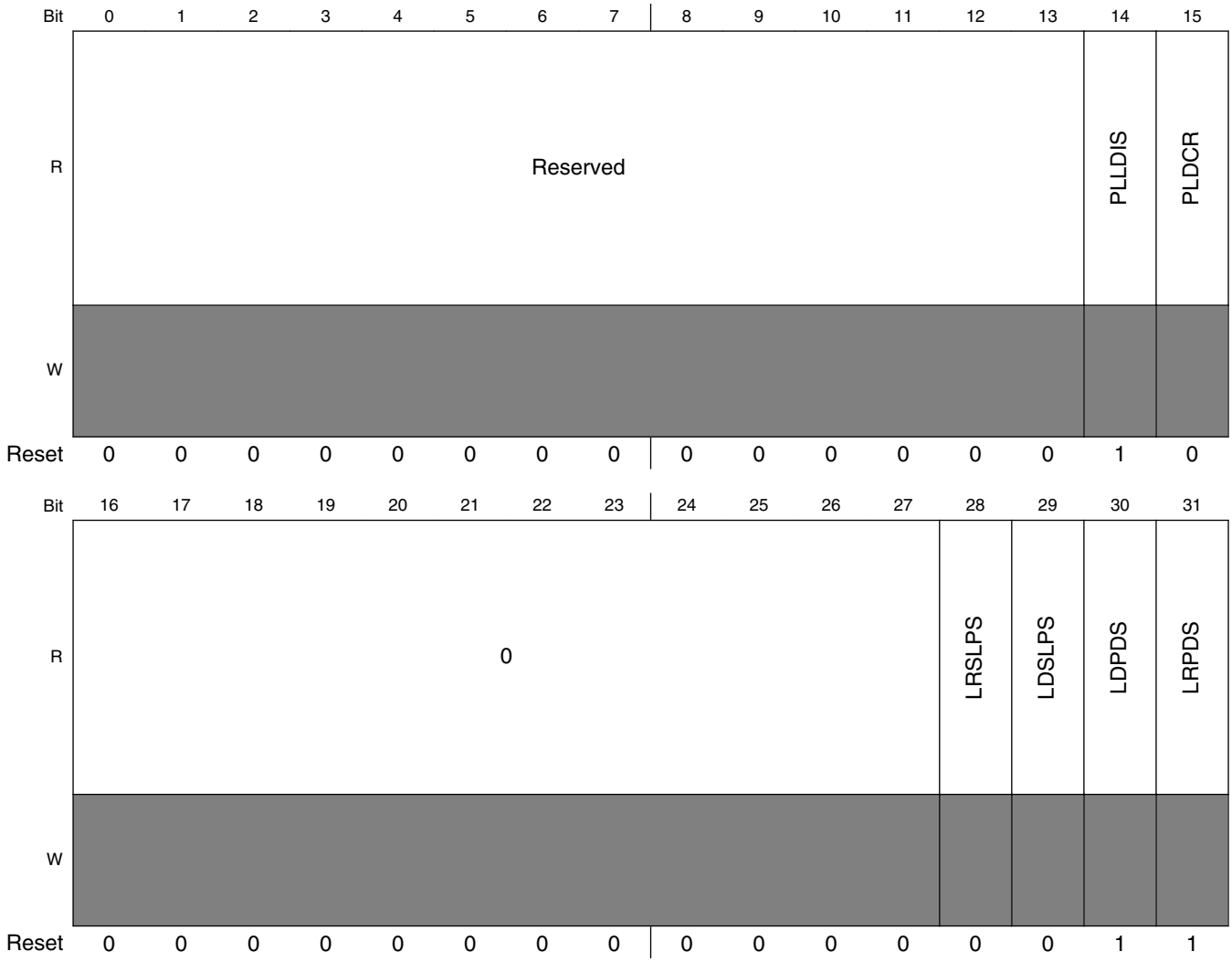
Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

52.6.24 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 0h base + A4h offset = A4h



LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

52.6.25 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0					URPCNT		
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

52.6.26 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)

Address: 0h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR*n* field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

52.7 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements[12]. All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

52.8 Functional description

12. See the “Functional Safety” chapter for register classification details.

52.8.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

52.8.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS parameters in the device data sheet.
2. The PLLCR is programmed with configuration parameters for the PLL.
3. The LCR is programed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN]and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurrence of one of the following:
    - LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
    - LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

14. Write PLLCR[SWPON] = 1 to enable the LFAST masters PLL.

15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:

Note

(The slaves Rx interface speed should be changed first.)

    - Write ICR[ICLCPLD] = 10h for Tx data fast frame.
    - Write ICR[SNDICLC] = 1.
19. Both of the following operations are performed to change the LFAST masters Rx interface speed:

Note

(The slaves Tx interface speed mode should be changed first)

    - Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
    - Write ICR[SNDICLC] = 1.
20. The ICLC transmission is confirmed by the occurrence of one of the following:
    - ICR[SNDICLC] = 0
    - TISR[TXICLCF] = 1.
21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
23. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
24. Write ICR[SNDICLC] = 1

25. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitation.
26. The LFAST slaves speed mode is confirmed by the occurrence of the following:
    - If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
    - If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.
27. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:
    - Write ICR[ICLCSEQ] = 0.
    - Write MCR[TXARBD] = 0.
28. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
    - Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
    - Write MCR[TXARBD] = 0, to enable frame transmission.
    - The operations from step 14 need to be executed again.

52.8.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS datasheet.
2. The PLLCR is programed with the configuration parameters for the PLL.
3. The LCR is programed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signaland LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following are true:
    - When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
    - MCRMCR[TXEN] = 1.

8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received. H/W writes [ICRPF] = 1.
    - PICR[PNGREQ] = 1.

Speed mode change:

9. The PLL will start when one of the following conditions is met:
    - When an ICLC frame with payload 02h is received. H/W writes RIISR[IPONF] = 1.
    - Write PLLCR[SWPON] = 1.
10. The speed of the Tx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
    - SCR[TDR] = 1, when SCR[DRMD] = 0.
11. The speed of the Rx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
    - SCR[RDR] = 1, when SCR[DRMD] = 0.
12. A Ping Frame is sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received. H/W writes RIISR[ICRPF] = 1.
    - Write PICR[PNGREQ] = 1.
    - Write PICR[PNGREQ] = .

52.8.2 Line Receiver

This section describes the Line Receiver.

52.8.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 52-4 shows the block diagram of Uplink Controller.

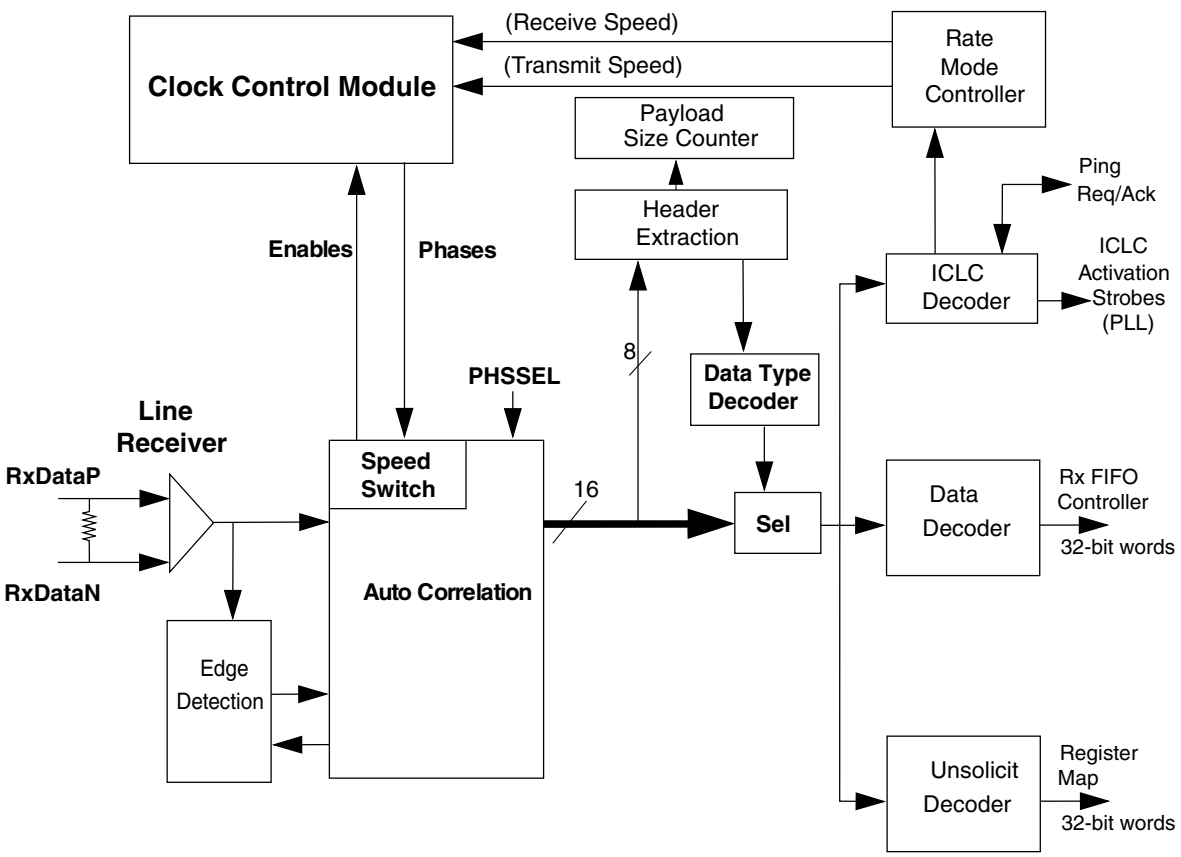


Figure 52-4. Top Level Receive Controller

52.8.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

52.8.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

52.8.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 52-5 shows the block diagram of Hunt Correlation mode.

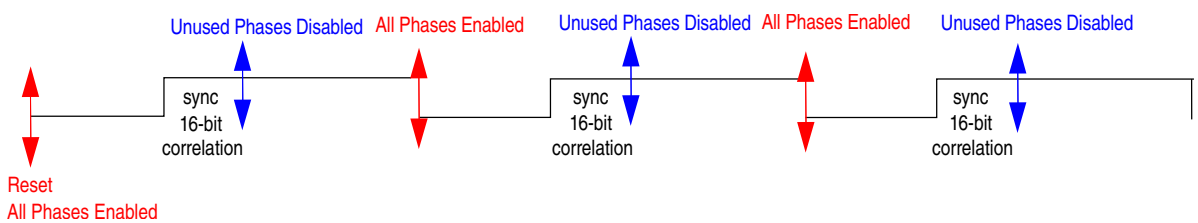


Figure 52-5. Hunt Correlation mode

52.8.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

52.8.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

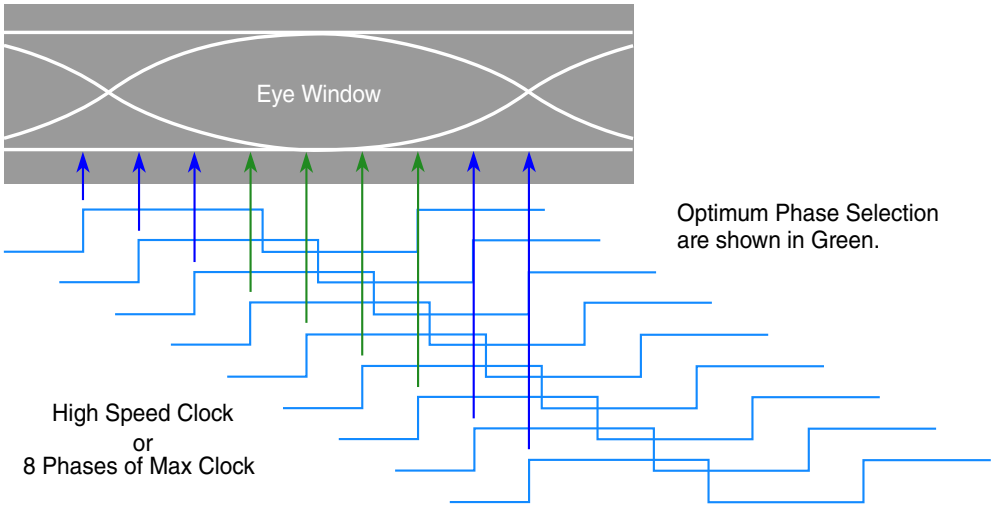


Figure 52-6. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

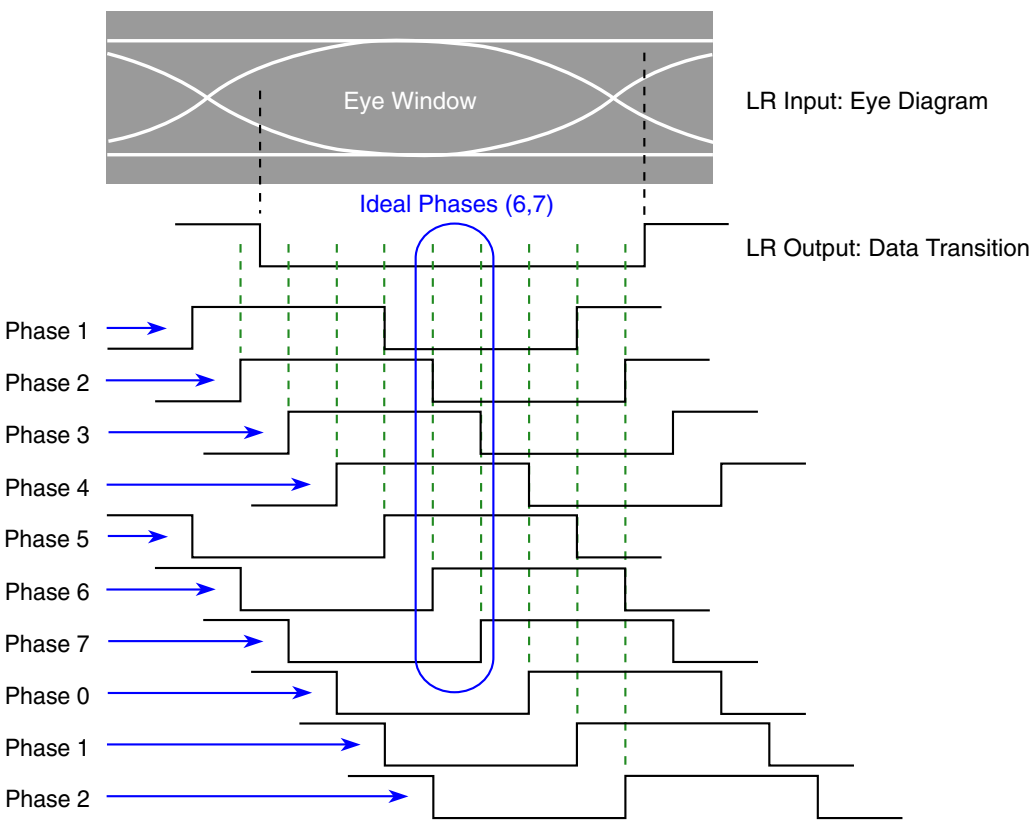


Figure 52-7. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in Figure 52-8 and Figure 52-9.

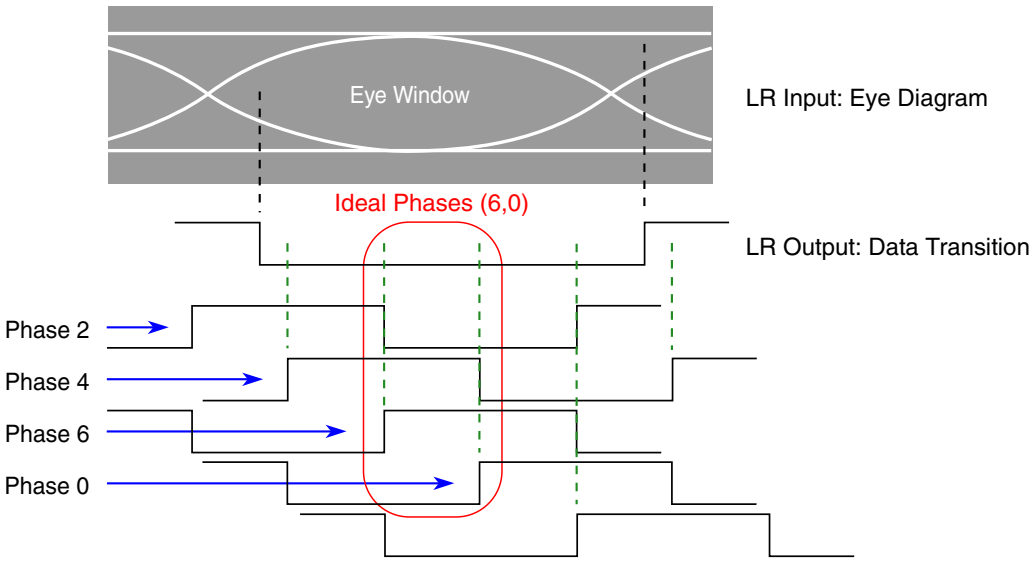


Figure 52-8. High Speed 4 Phase Clock Alignment Example

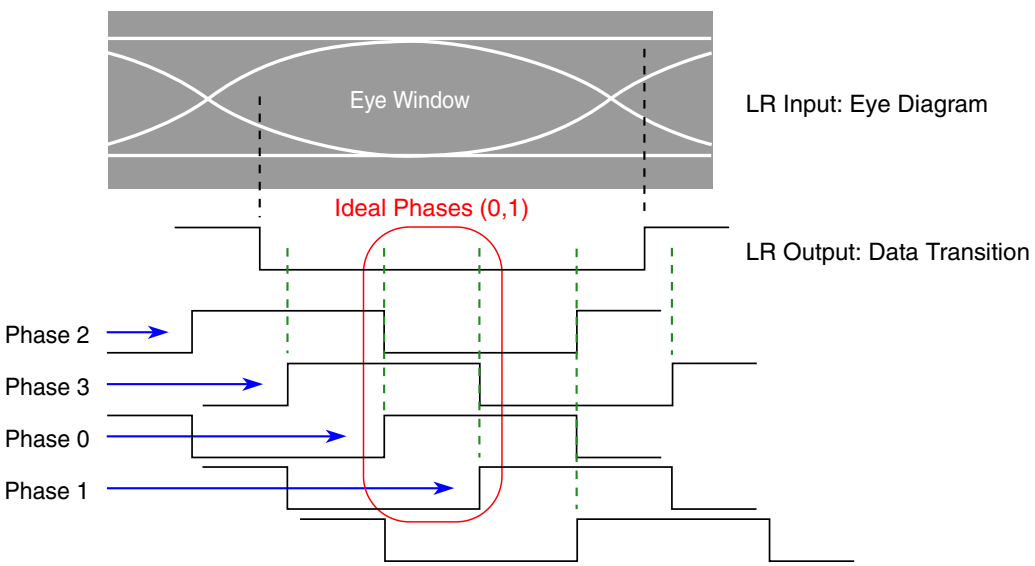


Figure 52-9. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in Table 52-2, Table 52-3, and Table 52-4 before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 52-2. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 52-3. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See Table 52-3, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 52-4. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 52-4. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See Table 52-4, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

52.8.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

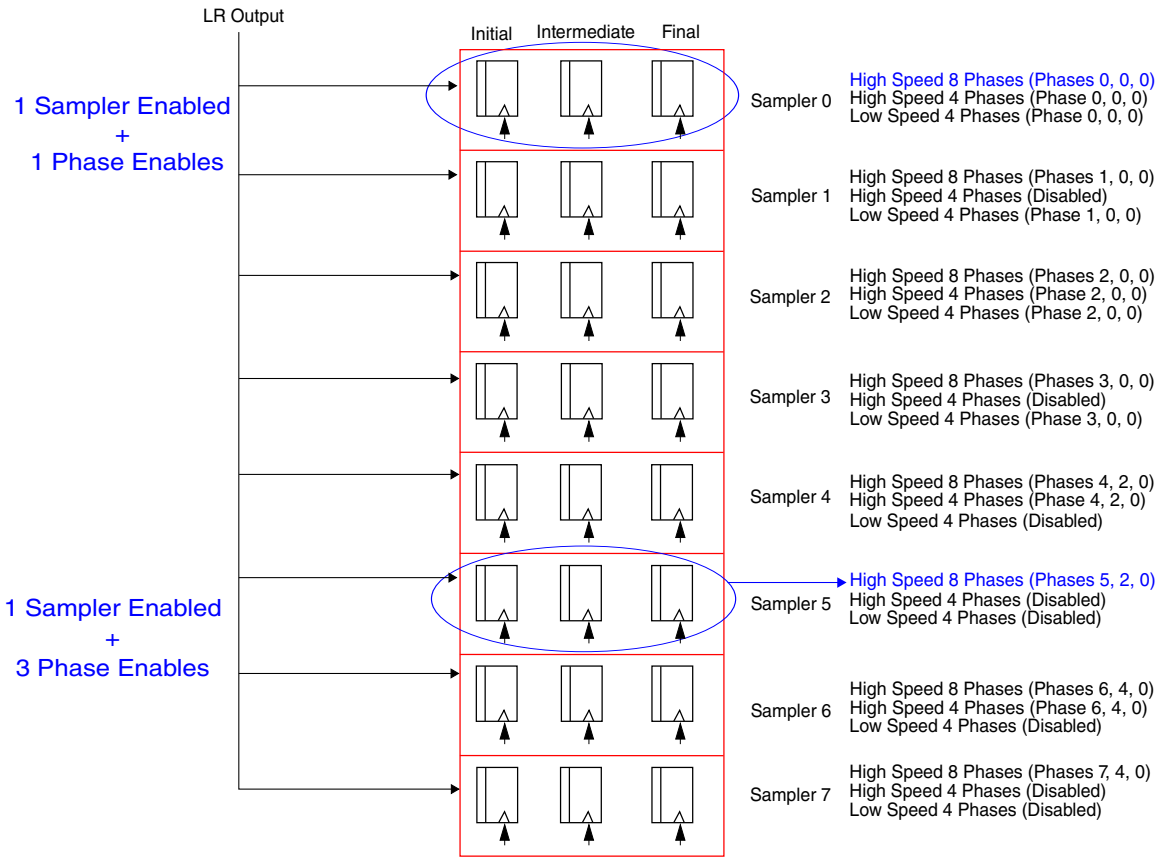


Figure 52-10. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

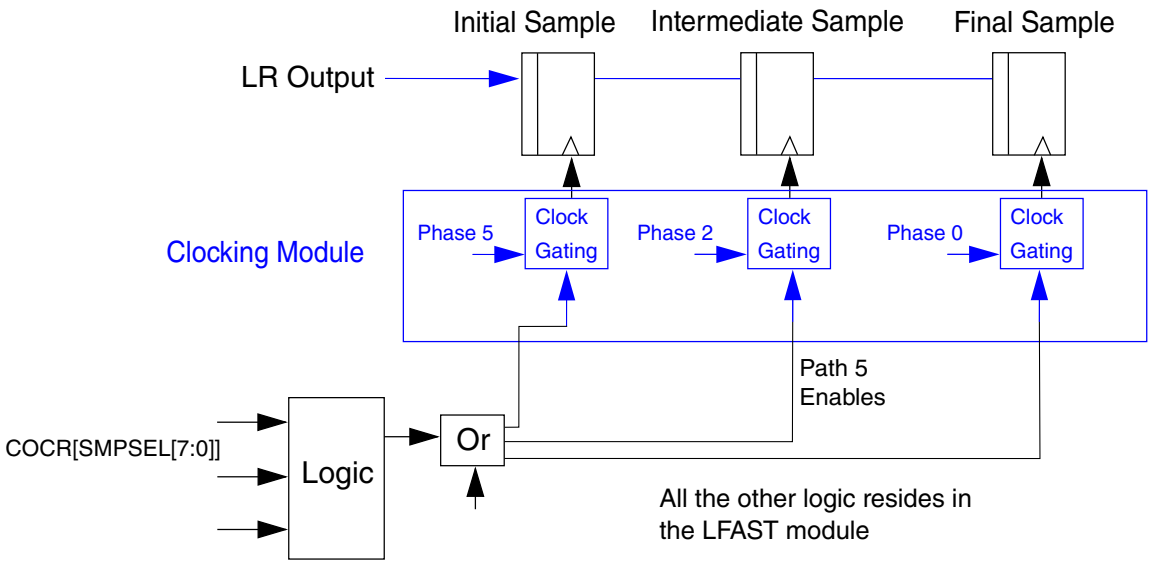


Figure 52-11. Sampler 5 Logic

52.8.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the Figure 52-12.

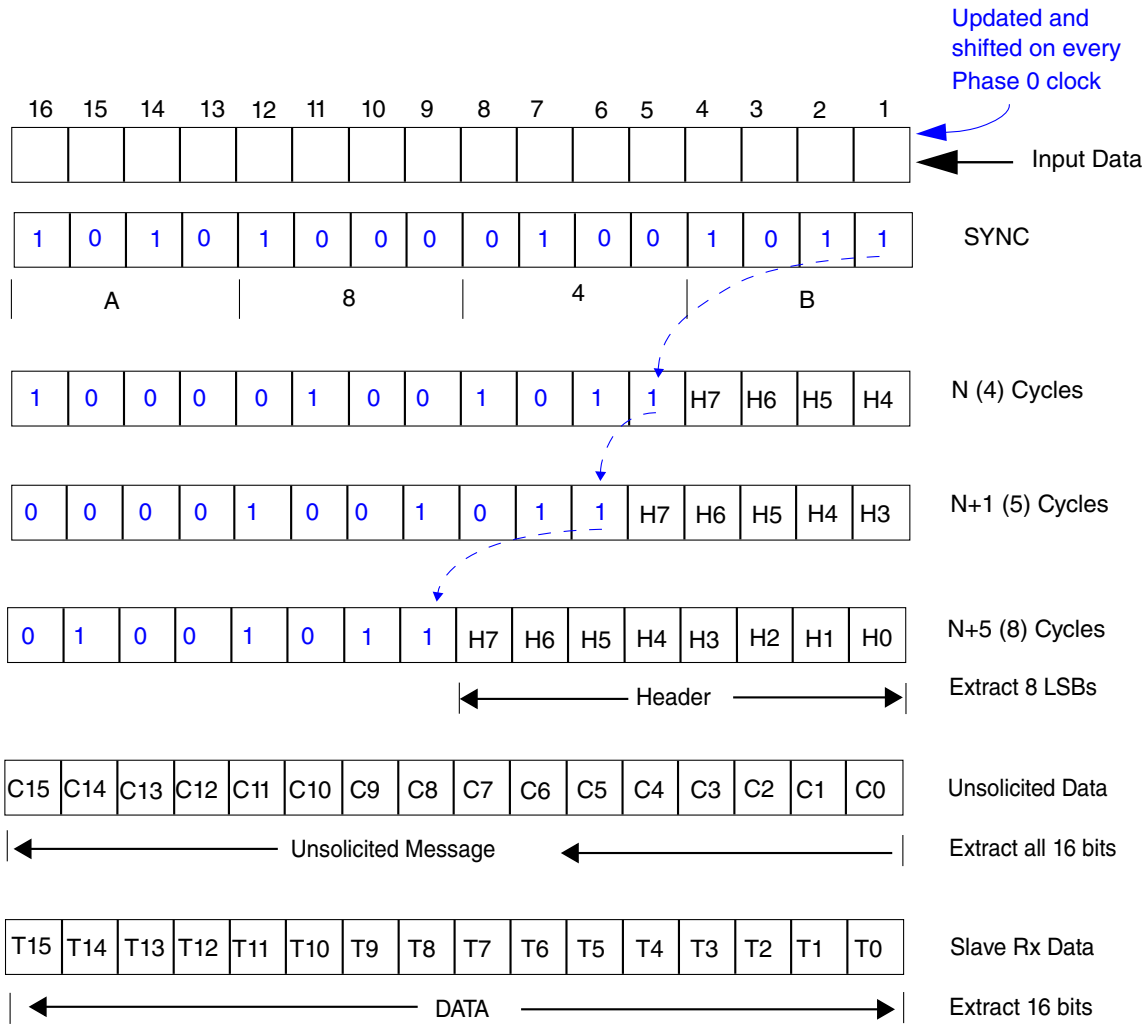


Figure 52-12. Extracting Header and Payload from the 16-bit output from Auto correlation.

52.8.3 Transmit Controller

This section describes the Transmit Controller.

52.8.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

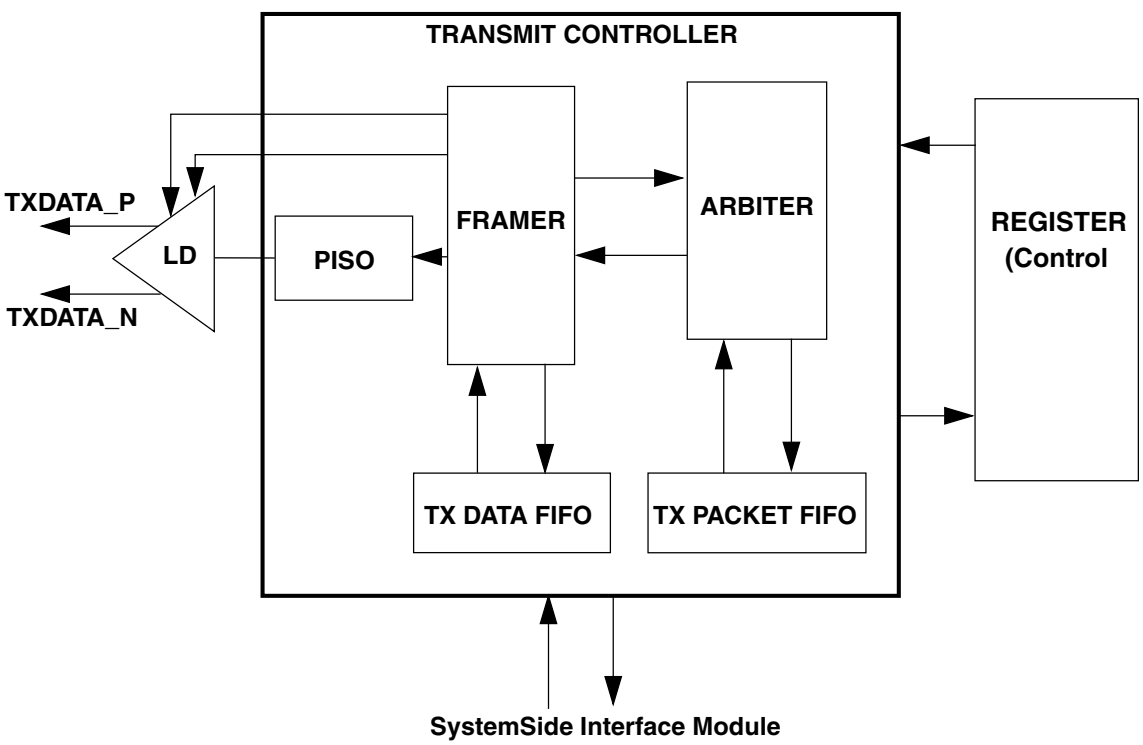


Figure 52-13. Transmit Controller Connections

52.8.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in Table 52-5

Table 52-5. Priority Levels for the Transmit Controller

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
LFAST interface enable	1	1	• LFAST interface enable is asserted • LFAST interface enable is negated
Tx Interface disabled	2	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	4	• TMCR[LPON] = 1Dig • RF Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	-	• LFAST Master: ICR[SNDICLC] = 1 • LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	-	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1

Table continues on the next page...

Table 52-5. Priority Levels for the Transmit Controller (continued)

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
Unsolicited frame request	6	6	• Last Frame with CTS = 1 received from the peer LFAST device • UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	• Tx FIFO contains one or more frames • Last Frame with CTS = 1 received from the peer LFAST device • MCR[DATAEN] = 1
CTSFrame	8	8	• MCR[CTSEN] = 1 • Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	• TMCR[CLKTST] = 0 • LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

52.8.3.3 Line Driver digital connections

52.8.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

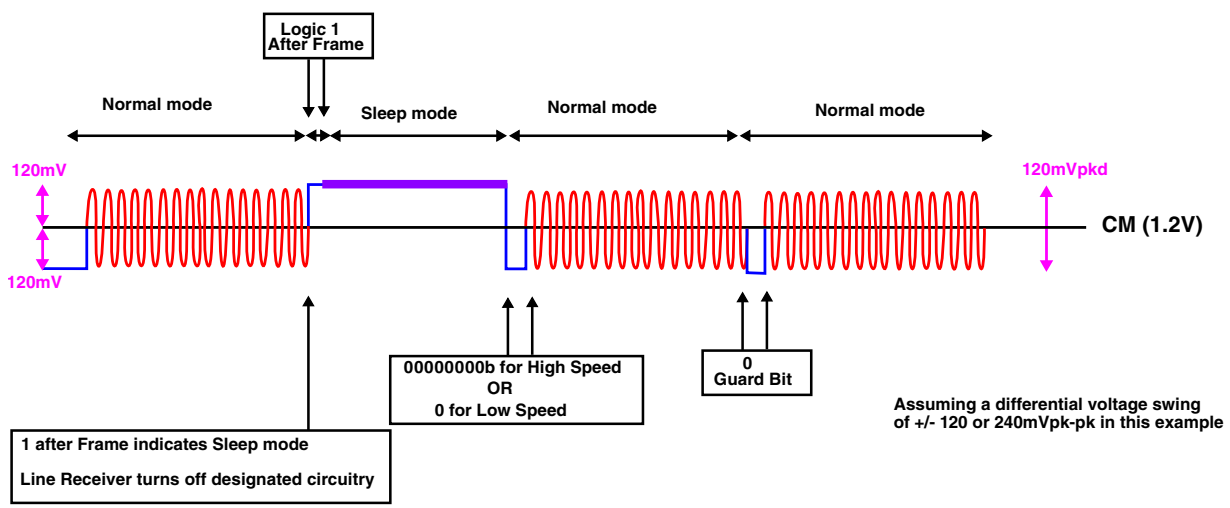


Figure 52-14. Example of Sleep mode

As shown in Figure 52-14 before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated,

and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 52-6. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	• Programing LVDS[SWOFFLD] and LVDS[SWONLD] • Programing MCR[TXEN] and MCR[DRFEN]	• LFAST interface enable negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	• No pending frame for transmission • Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

52.8.4 CTS mode support

LFAST module supports flow control methods. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device doesn't want the peer device to send data. The CTS bit of all frame are sent 0 untill the Rx FIFO pointer reaches the lower threshold, as described below.

- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 untill the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frame sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed untill a frame with CTS bit 1 is received.

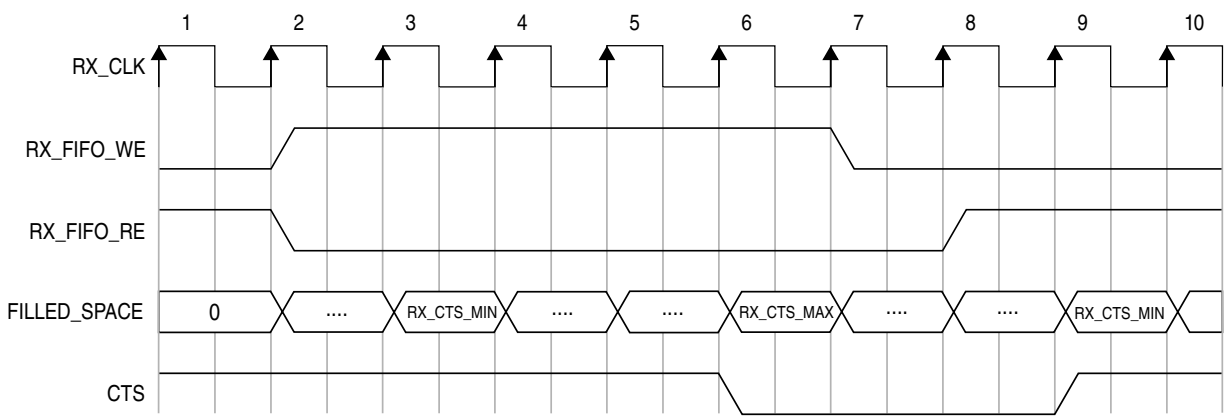


Figure 52-15. CTS generation

52.8.5 Frames supported

Table 52-7 provides the frames supported and their permissible payload sizes

Table 52-7. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO[1]	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in Table 52-7 then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

52.8.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

52.8.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

52.8.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and

schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

52.8.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

52.8.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

52.8.6.2 Unsolicited flow

52.8.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

52.8.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

52.8.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.

3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

52.8.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

52.8.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see Table 52-8 for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 52-8. Supported ICLC Payloads

ICLC Code(hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLLIn preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010… continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional)
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the Table 52-8.
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exist.

Note

ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in Table 52-8.

52.8.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in Table 52-8 will cause RISR[RXICF] = 1.

52.8.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

52.8.6.3.2.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

52.8.6.3.2.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

52.8.6.3.2.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

52.8.6.3.2.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

52.8.6.3.2.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

52.8.6.3.2.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable.

52.8.6.3.2.8 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

52.8.6.4 CTS flow

52.8.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

52.8.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frame with LCT type CTS are not stored in the LFAST.

52.8.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

52.8.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 52-9. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.

- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

52.8.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

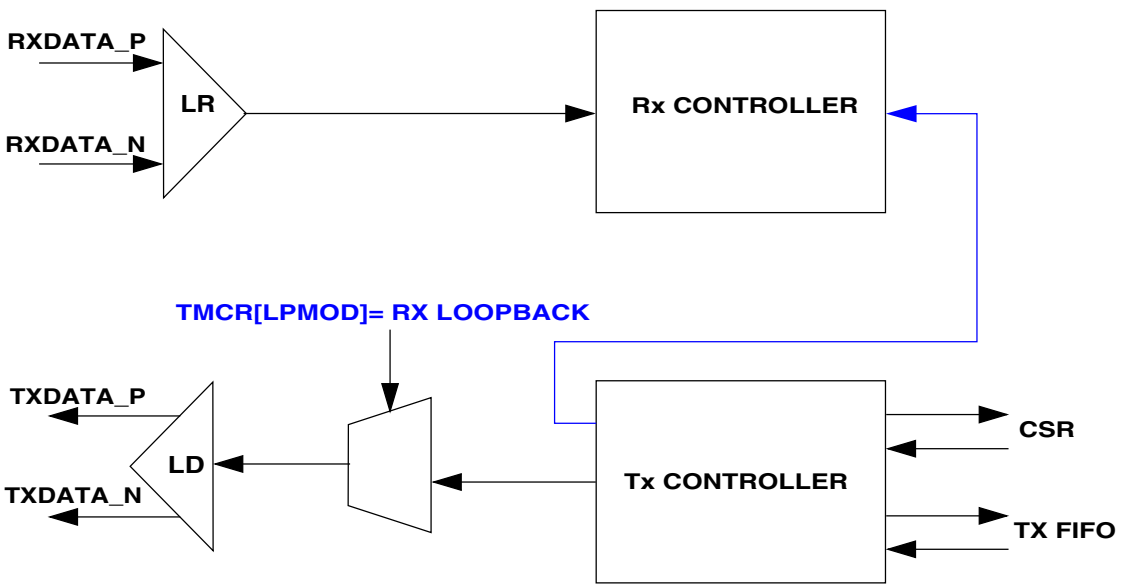


Figure 52-16. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

52.8.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

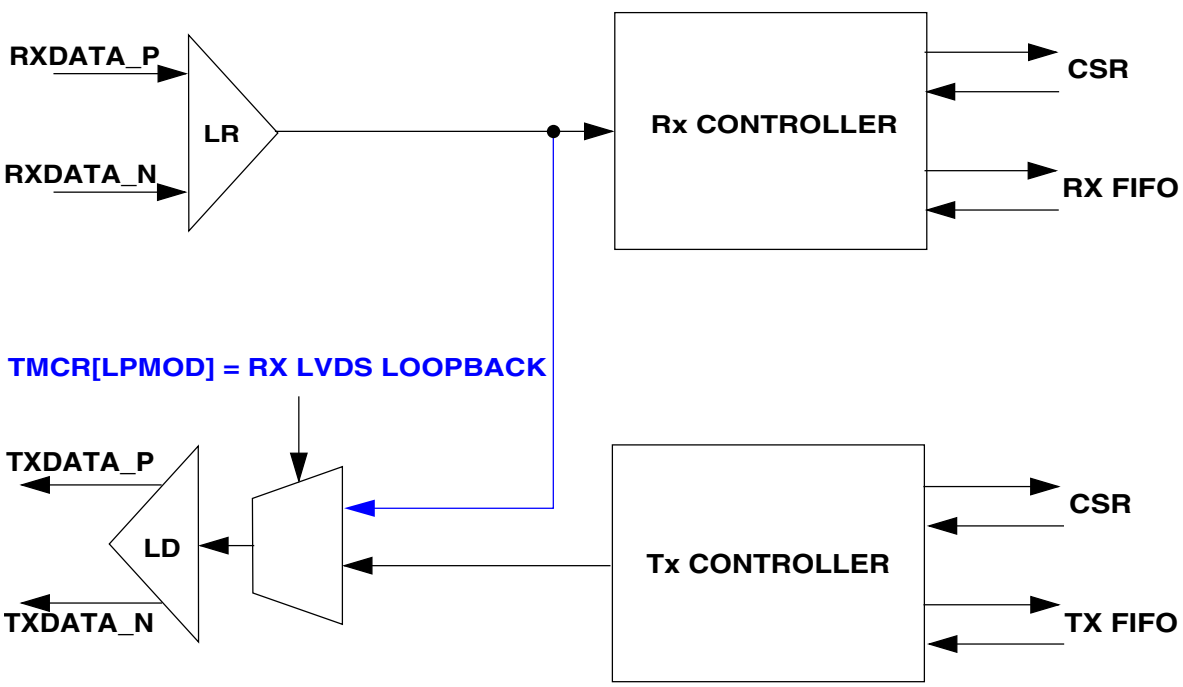


Figure 52-17. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:

- S/W writes TMCR[LPON] = 1.
- For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

52.8.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

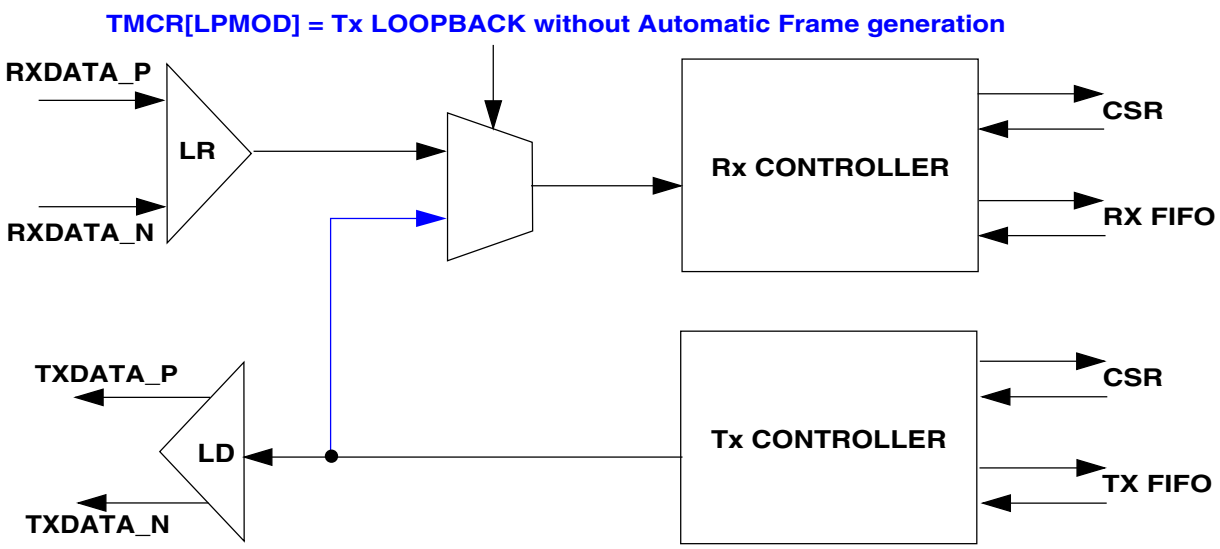


Figure 52-18. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

52.8.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

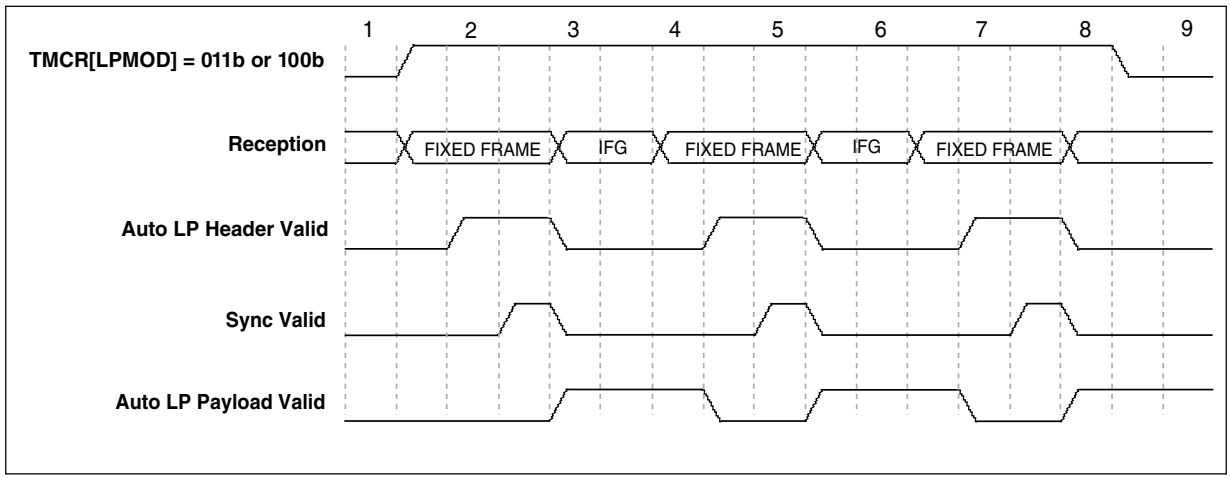


Figure 52-19. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

52.8.7.1.4.1 Tx loopback mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

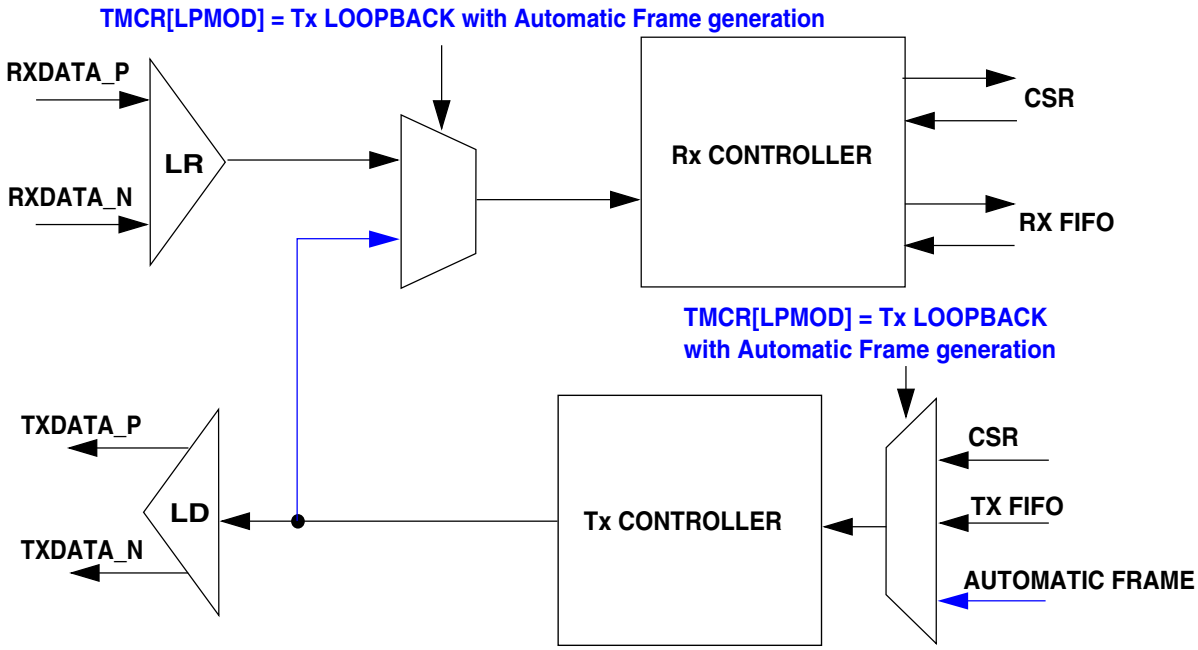


Figure 52-20. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

52.8.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

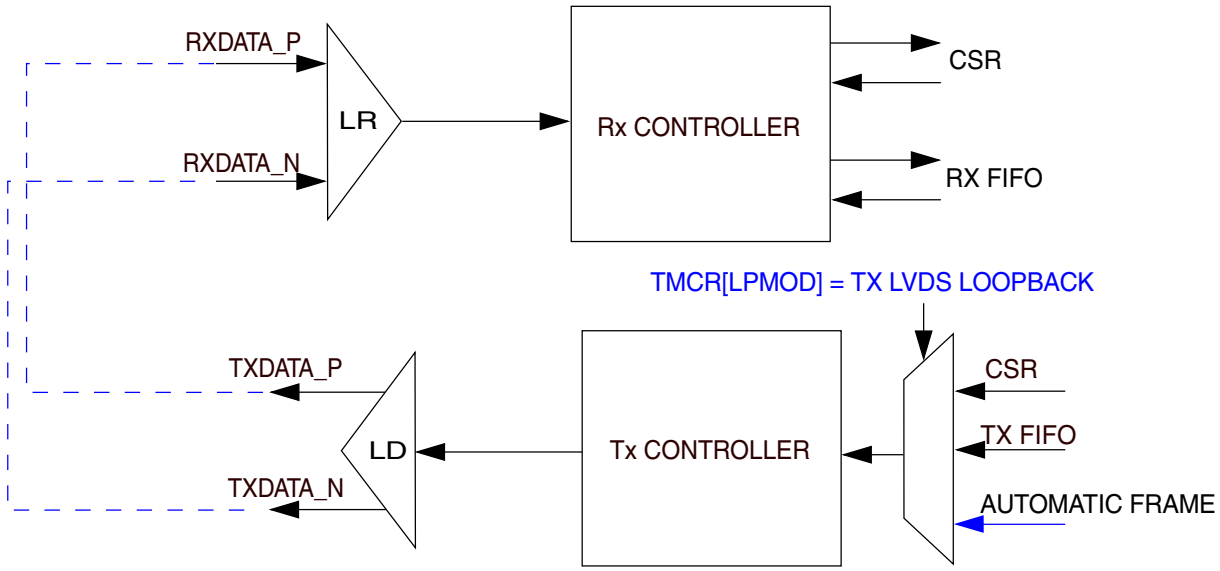


Figure 52-21. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

52.8.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

52.8.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC recieve interrupt

Each interrupt is asserted when any one of their status conditions is met.

52.8.8.1 Transmit complete interrupt

This interrupt indicates a transfer compeletion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

52.8.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 52-10. Recommended transmit exception handling

Exception	Recommendation
TXOVF	• MCR[TXEN] = 0 • The system level interfaces transmit is reseted and enabled again • For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set • For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	• The MCR[TXEN] should be set

52.8.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

52.8.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 52-11. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	• The MCR[RXEN] may be cleared • The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

52.8.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

52.9 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 52-12. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet[1]	1 × 8 max 1 packet[2]	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1
2. Only for ping response data when MCR[MSEN] = 1

52.10 Resets

The various blocks in LFAST are reset as described in the table below.

Table 52-13. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]

52.11 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

52.11.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

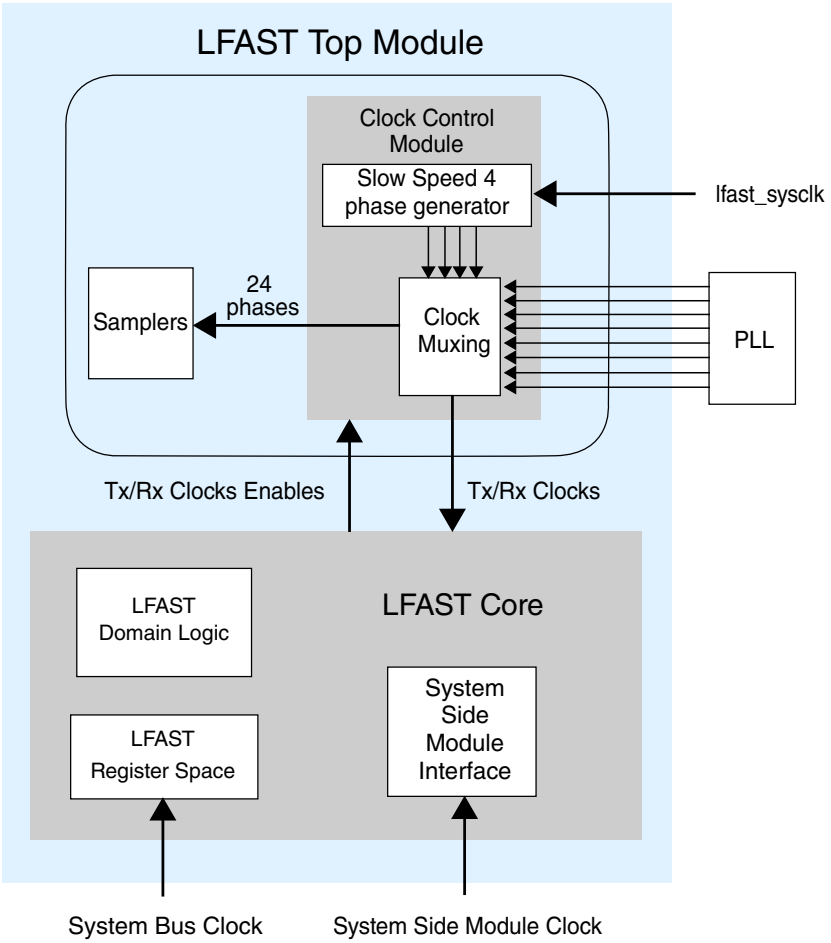


Figure 52-22. LFAST module clock domains

52.11.2 Slow speed clock

52.11.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in Figure 52-23, Figure 52-24 and Figure 52-25.

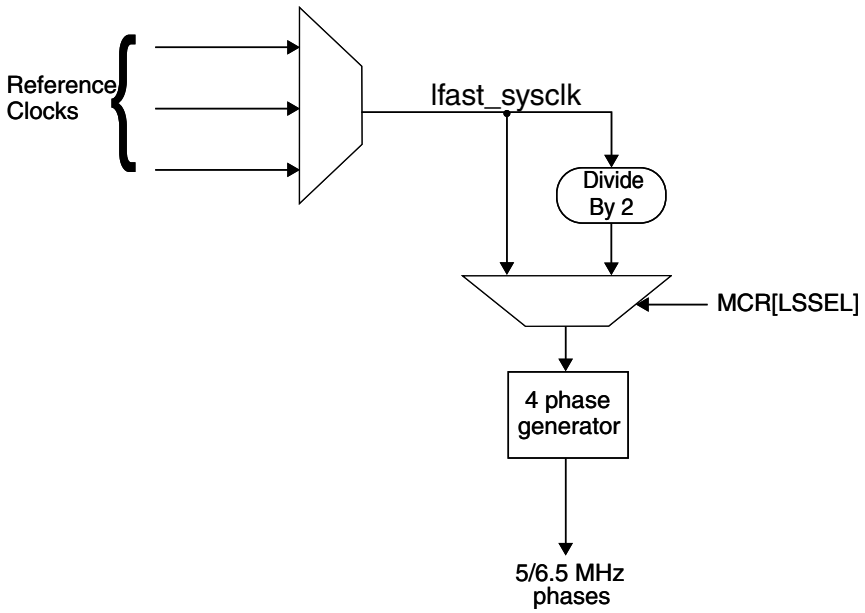


Figure 52-23. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in Figure 52-23.

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 52-24 (selected clock path shown in green).

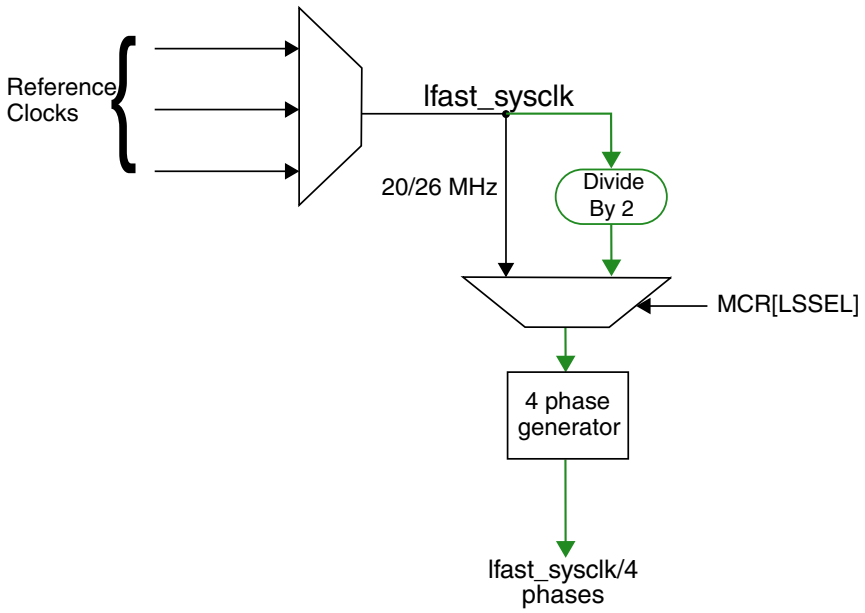


Figure 52-24. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 52-25 (selected clock path selected shown in magenta).

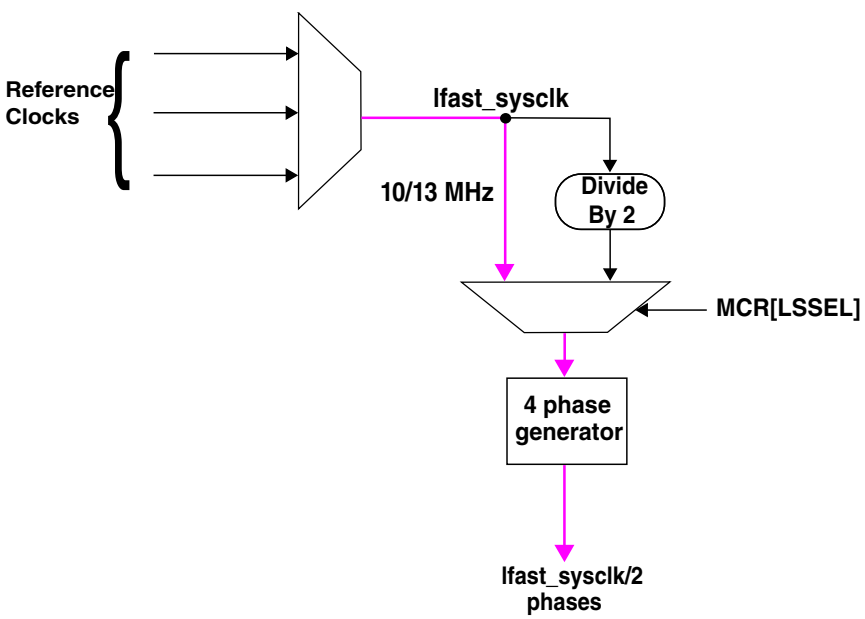


Figure 52-25. External clock muxing of lfast_sysclk of 10 MHz

52.11.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

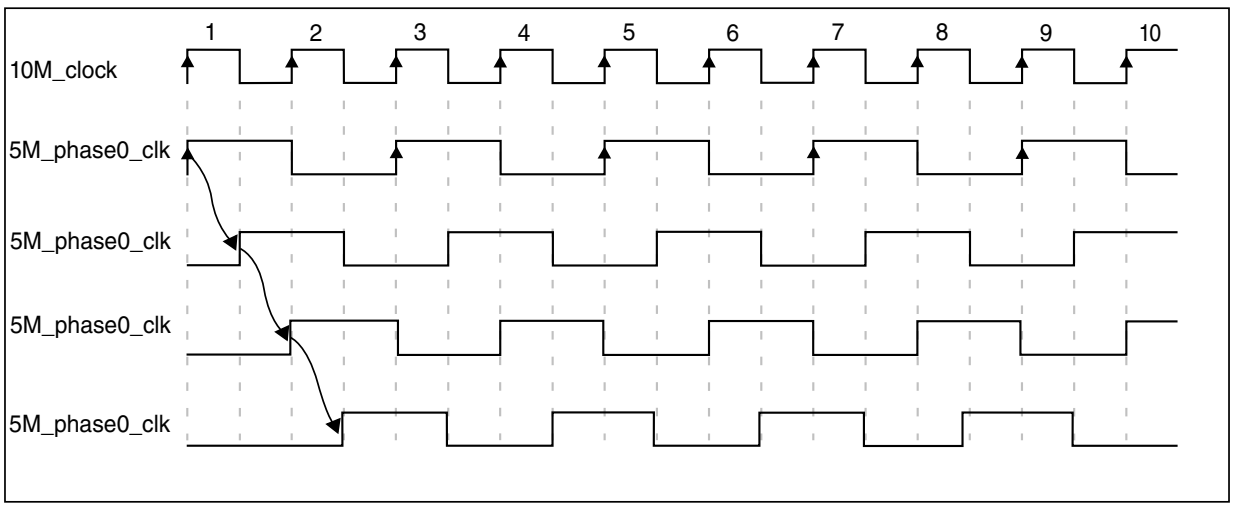


Figure 52-26. Slow speed 4 phases generation

52.11.3 Rx Controller Clocks

52.11.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

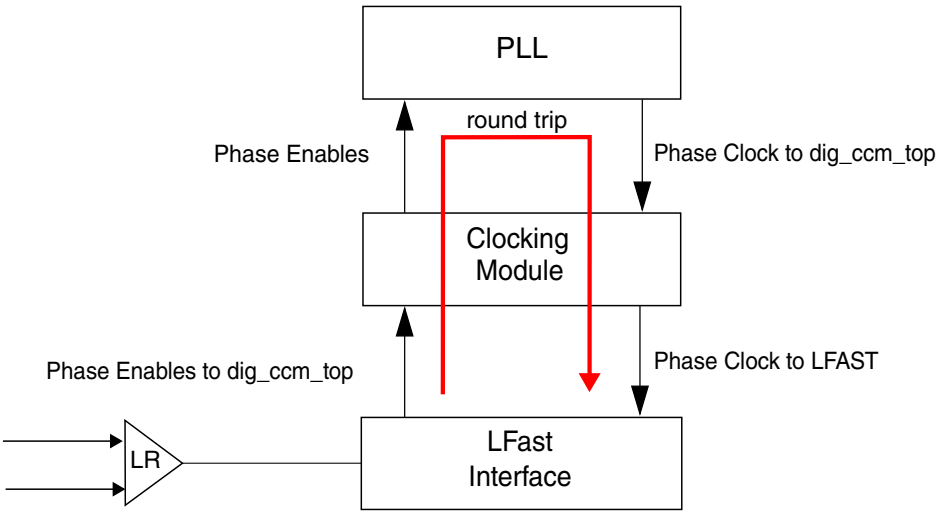


Figure 52-27. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

52.11.5 Clock module requirements for low speed phases

52.11.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 52-14. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

52.11.6 Tx Controller Clocks

52.11.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 52-15. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

52.11.6.2 Transmit Clock Muxing

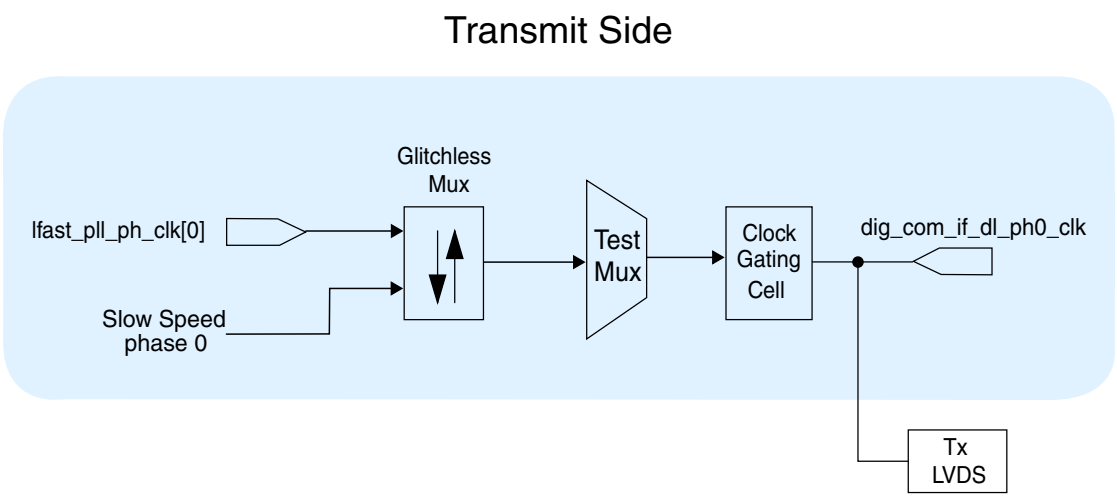


Figure 52-28. Transmit Clocks Muxing

52.11.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

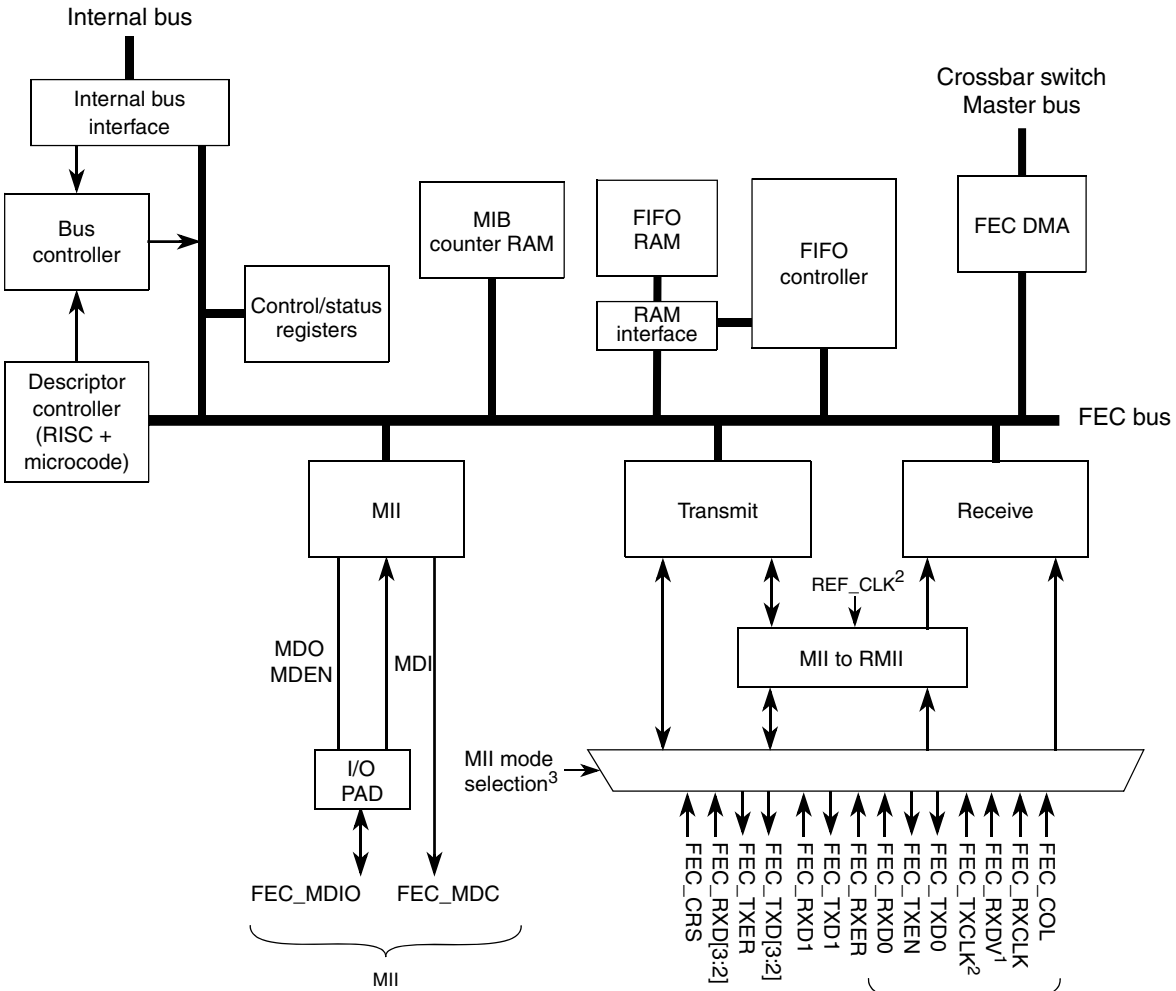
Chapter 53
Fast Ethernet Controller (FEC)

53.1 Introduction

The Fast Ethernet Controller (FEC) is a communication controller that supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports five different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII, 10/100 Mbps reduced MII, and the 10 Mbps-only 7-wire interface.

53.1.1 Block diagram

Figure 53-1 shows the block diagram of the FEC. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.



Notes:
1. FEC_RXDV in RMII mode generates RXDV and CRS.
2. FEC_TXCLK in RMII mode is the reference clock for the RMII; the RMII supplies the FEC's receive and transmit clocks depending upon the assertion of RCR[RMII_10T].
3. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

Figure 53-1. FEC block diagram

The following table provides a brief description of the blocks in the previous figure.

Table 53-1. FEC sub-block descriptions

Block	Description
Control and status registers (CSR)	Provides global control (Ethernet reset and enable) and interrupt managing registers
Descriptor controller	A RISC-based controller providing these functions in the FEC: • Initialization (those internal registers not initialized by you or hardware) • High-level control of the DMA channels (initiating DMA transfers) • Interpreting buffer descriptors

Table continues on the next page...

Table 53-1. FEC sub-block descriptions (continued)

Block	Description
	• Address recognition for receive frames • Random number generation for transmit collision backoff timer
DMA block	Provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to all run independently **Note:** In this document, "DMA" refers to the DMA block within the FEC, and is not related to any other DMA controllers that may be present on the chip.
Media independent interface (MII)	Provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of: • The management data clock (FEC_MDC) • The management data input/output lines (FEC_MDIO) of the MII interface
Message information block (MIB)	Maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC, but provides valuable counters for network management. The counters supported are: • The RMON (RFC 1757) Ethernet Statistics group • Some of the IEEE 802.3 counters
RAM	Serves as the focal point of all data flow in the Fast Ethernet controller and divides into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block. Receive data flows from the receive block into the receive FIFO.
Transmit and receive blocks	Provide the Ethernet MAC functionality (with some assist from microcode)

53.1.2 Features

The FEC incorporates the following features:

- Support for the following Ethernet physical interfaces:[1]
  - 100-Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII
  - 100-Mbps reduced media independent interface (RMII)
  - 10-Mbps reduced media independent interface (RMII)
  - 10-Mbps 7-wire interface (industry standard)

1. For interface selection options, see the chip-specific FEC information.

- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Support for half-duplex operation (100 Mbps throughput) with a minimum internal bus clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
    - Frames with broadcast address may be always accepted or always rejected
    - Exact match for single 48-bit individual (unicast) address
    - Hash (64-bit hash) check of individual (unicast) addresses
    - Hash (64-bit hash) check of group (multicast) addresses
    - Promiscuous mode

53.1.3 Clocking requirements

The FEC system clock frequency must be equal to or greater than the FEC protocol reference clock frequency (for example, 25 MHz system frequency for 100 Mbps protocol frequency). This requirement is true regardless of the reference selected by the application.

53.2 Modes of operation

This section describes the FEC operating modes.

53.2.1 Full and half duplex operation

Full duplex mode is for use on point-to-point links between switches or end node to switch. Half duplex mode works in connections between an end node and a repeater or between repeaters. TCR[FDEN] controls duplex mode selection.

When configured for full duplex mode, flow control may be enabled. For more details, see:

- The RFC_PAUSE and TFC_PAUSE fields in the TCR
- The RCR[FCE] field
- Full duplex flow control

53.2.2 Interface options

The following interface options are supported[2]. A detailed discussion of the interface configurations is provided in Network interface options.

53.2.2.1 10 Mbps and 100 Mbps MII interface

The IEEE 802.3 standard defines the media independent interface (MII) for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by programming RCR[MII_MODE]=1.

The operation speed is determined by the FEC_TXCLK and FEC_RXCLK pins driven by the external transceiver. The transceiver auto-negotiates the speed or software controls it via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions, as well as the section on the MII, for a description of how to read and write registers in the transceiver via this interface.

53.2.2.2 10 Mbps and 100 Mbps RMII interface

The reduced media independent interface (RMII) is a low-cost alternative to the IEEE802.3 MII standard. This interface provides the functionality of the MII interface on a total of 8 pins instead of 18. The RMII interface for 10/100 Ethernet MAC-PHY interface was defined by an industry consortium and is not currently included in the IEEE 802.3 standard. This functionality is selected as described in the chip configuration chapter, and is reflected in the read-only RCR[RMII_MODE] bit. The RCR[RMII_10T]

2. For interface selection options, see the chip-specific FEC information.

bit determines the speed of operation. The reference clock for RMII is always 50 MHz, but this clock can be divided by 10 within the RCR register to support 10 Mbps operation. The PHY must be configured accordingly.

53.2.2.3 10 Mbps 7-wire interface operation

The FEC supports 7-wire interface used by many 10 Mbps Ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, MII mode is disabled and the 10 Mbps 7-wire mode is enabled.

53.2.3 Address recognition options

The address options supported are:

- Promiscuous
- Broadcast reject
- Individual address (hash or exact match)
- Multicast hash match

Address recognition options are discussed in detail in Ethernet address recognition.

53.2.4 Internal loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in MII internal and external loopback.

53.3 External signal description

The following table describes the various FEC signals, as well as indicating which signals work in available modes.

Table 53-2. FEC signal descriptions

Signal name	MII	7-wire	RMII	Description
FEC_COL	X	X	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.
FEC_CRS	X	—	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle.

Table continues on the next page...

Table 53-2. FEC signal descriptions (continued)

Signal name	MII	7-wire	RMII	Description
				In RMII mode, this signal is present on the FEC_RXDV pin.
FEC_MDC	X	—	X	Output clock provides a timing reference to the PHY for data transfers on the FEC_MDIO signal.
FEC_MDIO	X	—	X	Transfers control information between the external PHY and the media-access controller. Data is synchronous to FEC_MDC. This signal is an input after reset. When the FEC operates in 10Mbps 7-wire interface mode, this signal should be connected to VSS.
FEC_RXCLK	X	X	—	Provides a timing reference for FEC_RXDV, FEC_RXD[3:0], and FEC_RXER.
FEC_RXDV	X	X	X	Asserting the FEC_RXDV input indicates PHY has valid nibbles present on the MII. FEC_RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of FEC_RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.
FEC_RXD0	X	X	X	This pin contains the Ethernet input data transferred from PHY to the media-access controller when FEC_RXDV is asserted.
FEC_RXD1	X	—	X	This pin contains the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXD[3:2]	X	—	—	These pins contain the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXER	X	—	X	When asserted with FEC_RXDV, indicates PHY detects an error in the current frame. When FEC_RXDV is not asserted, FEC_RXER has no effect.
FEC_TXCLK	X	X	X	Input clock which provides a timing reference for FEC_TXEN, FEC_TXD[3:0] and FEC_TXER. In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.
FEC_TXD0	X	X	X	The serial output Ethernet data and only valid during the assertion of FEC_TXEN.
FEC_TXD1	X	—	X	This pin contains the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXD[3:2]	X	—	—	These pins contain the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXEN	X	X	X	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first FEC_TXCLK following the final nibble of the frame.
FEC_TXER	X	—	—	When asserted for one or more clock cycles while FEC_TXEN is also asserted, PHY sends one or more illegal symbols. FEC_TXER has no effect at 10 Mbps or when FEC_TXEN is negated.
Note: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.				

53.4 Memory map and register definition

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs control operation modes and extract global status information. The descriptors pass data buffers and related buffer information between the hardware and software.

Each FEC implementation requires a 1 KB memory map space, which is divided into two sections of 512 bytes each for:

- Control/status registers
- Event/statistic counters held in the MIB block

The following table defines the top-level memory map.

Table 53-3. FEC memory map

Address offset	Function
000h–1FFh	Control/status registers
200h–3FFh	MIB block counters

The following table shows the FEC register map. Any address offset not explicitly mentioned in this table is reserved. Do not modify the content of reserved fields. All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit. Accessing reserved or unimplemented register spaces in the FEC will not return transfer errors.

The MIB counters (shown in the following table) define the locations in the MIB RAM space where hardware-maintained counters reside. The counters are divided into two groups:

- RMON counters include the Ethernet statistics counters defined in RFC 1757
- Truncated frames counter (only frames with lengths up to 2047 bytes are supported)

The transmit and receive RMON counters are independent, which ensures accurate network statistics when operating in full duplex mode.

The included IEEE counters support the mandatory and recommended counter packages defined in Section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The FEC supports IEEE Basic Package objects, but these do not require counters in the MIB block. In addition, some of the recommended package objects supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are also included.

FEC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Interrupt Event Register (FEC_EIR)	32	w1c	0000_0000h	53.4.1/2354
8	Interrupt Mask Register (FEC_EIMR)	32	R/W	0000_0000h	53.4.2/2356
10	Receive Descriptor Active Register (FEC_RDAR)	32	R/W	0000_0000h	53.4.3/2358
14	Transmit Descriptor Active Register (FEC_TDAR)	32	R/W	0000_0000h	53.4.4/2359
24	Ethernet Control Register (FEC_ECR)	32	R/W	F000_0000h	53.4.5/2360
40	MII Management Frame Register (FEC_MMFR)	32	R/W	Undefined	53.4.6/2361
44	MII Speed Control Register (FEC_MSCR)	32	R/W	0000_0000h	53.4.7/2362
64	MIB Control Register (FEC_MIBC)	32	R/W	C000_0000h	53.4.8/2363
84	Receive Control Register (FEC_RCR)	32	R/W	05EE_0001h	53.4.9/2365
C4	Transmit Control Register (FEC_TCR)	32	R/W	0000_0000h	53.4.10/ 2367
E4	Physical Address Low Register (FEC_PALR)	32	R/W	See section	53.4.11/ 2368
E8	Physical Address High Register and Type Field (FEC_PAUR)	32	R/W	See section	53.4.12/ 2369
EC	Opcode/Pause Duration (FEC_OPD)	32	R/W	See section	53.4.13/ 2369
118	Descriptor Individual Upper Address Register (FEC_IAUR)	32	R/W	See section	53.4.14/ 2370
11C	Descriptor Individual Lower Address Register (FEC_IALR)	32	R/W	See section	53.4.15/ 2370
120	Descriptor Group Upper Address Register (FEC_GAUR)	32	R/W	See section	53.4.16/ 2371
124	Descriptor Group Lower Address Register (FEC_GALR)	32	R/W	See section	53.4.17/ 2371
144	Transmit FIFO Watermark (FEC_TFWR)	32	R/W	0000_0000h	53.4.18/ 2372
14C	FIFO Receive Bound Register (FEC_FRBR)	32	R	0000_0600h	53.4.19/ 2373
150	FIFO Receive Start Register (FEC_FRSR)	32	R/W	0000_0500h	53.4.20/ 2373
180	Receive Descriptor Ring Start Register (FEC_ERDSR)	32	R/W	See section	53.4.21/ 2374
184	Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)	32	R/W	See section	53.4.22/ 2375
188	Receive Buffer Size Register (FEC_EMRBR)	32	R/W	See section	53.4.23/ 2375
200	Count of frames not counted correctly (FEC_RMON_T_DROP)	32	R/W	Undefined	53.4.24/ 2376
204	RMON Tx packet count (FEC_RMON_T_PACKETS)	32	R/W	Undefined	53.4.25/ 2377
208	RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)	32	R/W	Undefined	53.4.26/ 2377

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C	RMON Tx multicast packets (FEC_RMON_T_MC_PKT)	32	R/W	Undefined	53.4.27/ 2378
210	RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN)	32	R/W	Undefined	53.4.28/ 2378
214	RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE)	32	R/W	Undefined	53.4.29/ 2379
218	RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE)	32	R/W	Undefined	53.4.30/ 2379
21C	RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG)	32	R/W	Undefined	53.4.31/ 2380
220	RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB)	32	R/W	Undefined	53.4.32/ 2380
224	RMON Tx collision count (FEC_RMON_T_COL)	32	R/W	Undefined	53.4.33/ 2381
228	RMON Tx 64 byte packets (FEC_RMON_T_P64)	32	R/W	Undefined	53.4.34/ 2381
22C	RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127)	32	R/W	Undefined	53.4.35/ 2382
230	RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255)	32	R/W	Undefined	53.4.36/ 2382
234	RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511)	32	R/W	Undefined	53.4.37/ 2383
238	RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023)	32	R/W	Undefined	53.4.38/ 2383
23C	RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047)	32	R/W	Undefined	53.4.39/ 2384
240	RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048)	32	R/W	Undefined	53.4.40/ 2384
244	RMON Tx Octets (FEC_RMON_T_OCTETS)	32	R/W	Undefined	53.4.41/ 2385
248	Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP)	32	R/W	Undefined	53.4.42/ 2385
24C	Frames transmitted OK (FEC_IEEE_T_FRAME_OK)	32	R/W	Undefined	53.4.43/ 2386
250	Frames transmitted with single collision (FEC_IEEE_T_1COL)	32	R/W	Undefined	53.4.44/ 2386
254	Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL)	32	R/W	Undefined	53.4.45/ 2387
258	Frames transmitted after deferral delay (FEC_IEEE_T_DEF)	32	R/W	Undefined	53.4.46/ 2387
25C	Frames transmitted with late collision (FEC_IEEE_T_LCOL)	32	R/W	Undefined	53.4.47/ 2388
260	Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL)	32	R/W	Undefined	53.4.48/ 2388

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
264	Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR)	32	R/W	Undefined	53.4.49/ 2389
268	Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR)	32	R/W	Undefined	53.4.50/ 2389
26C	Frames transmitted with SQE error (FEC_IEEE_T_SQE)	32	R/W	Undefined	53.4.51/ 2390
270	Flow control pause frames transmitted (FEC_IEEE_T_FDXFC)	32	R/W	Undefined	53.4.52/ 2390
274	Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK)	32	R/W	Undefined	53.4.53/ 2391
280	Count of received frames not counted correctly (FEC_RMON_R_DROP)	32	R/W	Undefined	53.4.54/ 2391
284	RMON Rx packet count (FEC_RMON_R_PACKETS)	32	R/W	Undefined	53.4.55/ 2392
288	RMON Rx broadcast packets (FEC_RMON_R_BC_PKT)	32	R/W	Undefined	53.4.56/ 2392
28C	RMON Rx multicast packets (FEC_RMON_R_MC_PKT)	32	R/W	Undefined	53.4.57/ 2393
290	RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN)	32	R/W	Undefined	53.4.58/ 2393
294	RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE)	32	R/W	Undefined	53.4.59/ 2394
298	RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE)	32	R/W	Undefined	53.4.60/ 2394
29C	RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)	32	R/W	Undefined	53.4.61/ 2395
2A0	RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)	32	R/W	Undefined	53.4.62/ 2395
2A4	Reserved (FEC_RMON_R_RESVD_0)	32	R/W	Undefined	53.4.63/ 2396
2A8	RMON Rx 64 byte packets (FEC_RMON_R_P64)	32	R/W	Undefined	53.4.64/ 2396
2AC	RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)	32	R/W	Undefined	53.4.65/ 2397
2B0	RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)	32	R/W	Undefined	53.4.66/ 2397
2B4	RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511)	32	R/W	Undefined	53.4.67/ 2398
2B8	RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023)	32	R/W	Undefined	53.4.68/ 2398
2BC	RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)	32	R/W	Undefined	53.4.69/ 2399
2C0	RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)	32	R/W	Undefined	53.4.70/ 2399

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2C4	RMON Rx octets (FEC_RMON_R_OCTETS)	32	R/W	Undefined	53.4.71/ 2400
2C8	Count of received frames not counted correctly (FEC_IEEE_R_DROP)	32	R/W	Undefined	53.4.72/ 2400
2CC	Frames received OK (FEC_IEEE_R_FRAME_OK)	32	R/W	Undefined	53.4.73/ 2401
2D0	Frames received with CRC error (FEC_IEEE_R_CRC)	32	R/W	Undefined	53.4.74/ 2401
2D4	Frames received with alignment error (FEC_IEEE_R_ALIGN)	32	R/W	Undefined	53.4.75/ 2402
2D8	Receive FIFO overflow count (FEC_IEEE_R_MACERR)	32	R/W	Undefined	53.4.76/ 2402
2DC	Flow control pause frames received (FEC_IEEE_R_FDXFC)	32	R/W	Undefined	53.4.77/ 2403
2E0	Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)	32	R/W	Undefined	53.4.78/ 2403

53.4.1 Interrupt Event Register (FEC_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0		
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_EIR field descriptions

Field	Description
0 HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
1 BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of these conditions: • A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. • A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. • A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. See Full duplex flow control.
4 TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
7 RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
8 MII	MII interrupt. Indicates the MII has completed the data transfer requested.
9 EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to ensure that the FIFO controller and DMA also soft reset.

Table continues on the next page...

FEC_EIR field descriptions (continued)

Field	Description
10 LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision retry limit. Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
12 UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.2 Interrupt Mask Register (FEC_EIMR)

The EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

Each bit in this register corresponds to an interrupt source defined by the EIR. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_EIMR field descriptions

Field	Description
0 HBERR	Heartbeat error interrupt mask.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The interrupt source is masked. 1 The interrupt source is not masked.
1 BABR	Babbling receive error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
2 BABT	Babbling transmit error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
3 GRA	Graceful stop complete interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
4 TXF	Transmit frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
5 TXB	Transmit buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
6 RXF	Receive frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
7 RXB	Receive buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
8 MII	MII interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
9 EBERR	Ethernet bus error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
10 LC	Late collision interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
11 RL	Collision retry limit interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
12 UN	Transmit FIFO underrun interrupt mask.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The interrupt source is masked. 1 The interrupt source is not masked.
13–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.3 Receive Descriptor Active Register (FEC_RDAR)

RDAR is a command register, written by the user, indicating the receive descriptor ring is updated (the driver produced empty receive buffers with the empty bit set).

When the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (if ECR[ETHER_EN] is also set). After the FEC polls a receive descriptor whose empty bit is not set, FEC clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring.

The RDAR is cleared at reset and when ECR[ETHER_EN] is cleared.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							RDAR	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_RDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.4 Transmit Descriptor Active Register (FEC_TDAR)

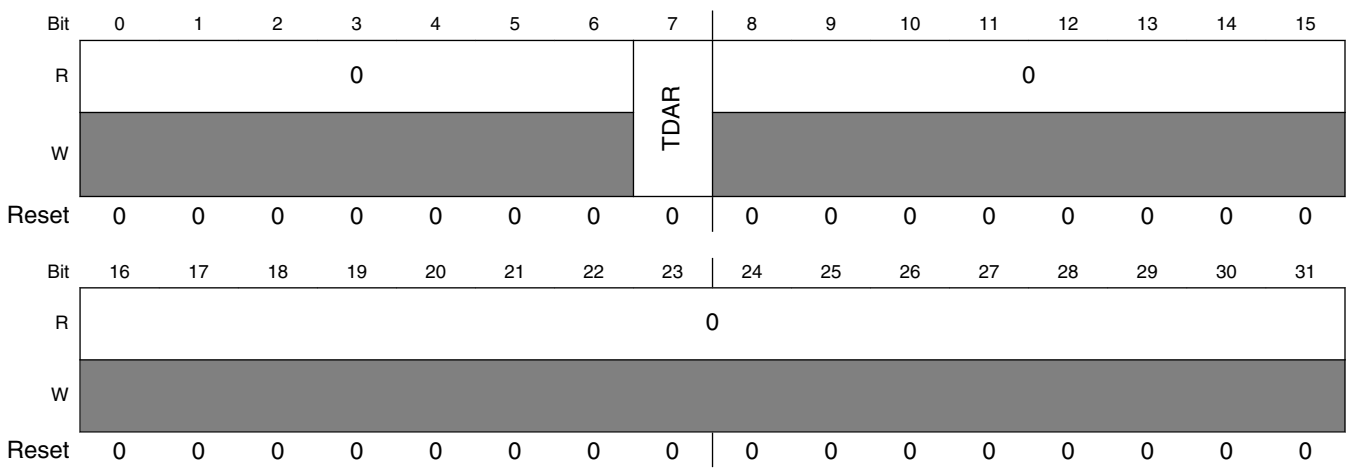
The TDAR is a command register which the user writes to indicate the transmit descriptor ring is updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

When the register is written, the TDAR field is set. This value is independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and processes transmit frames (provided ECR[ETHER_EN] is also set). After the FEC polls a transmit descriptor that is a ready bit not set, FEC clears the TDAR field and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors are placed into the transmit descriptor ring.

The TDAR field is cleared when any of the following events occur:

- Chip reset
- When ECR[ETHER_EN] is cleared
- When ECR[RESET] is set

Address: 0h base + 14h offset = 14h



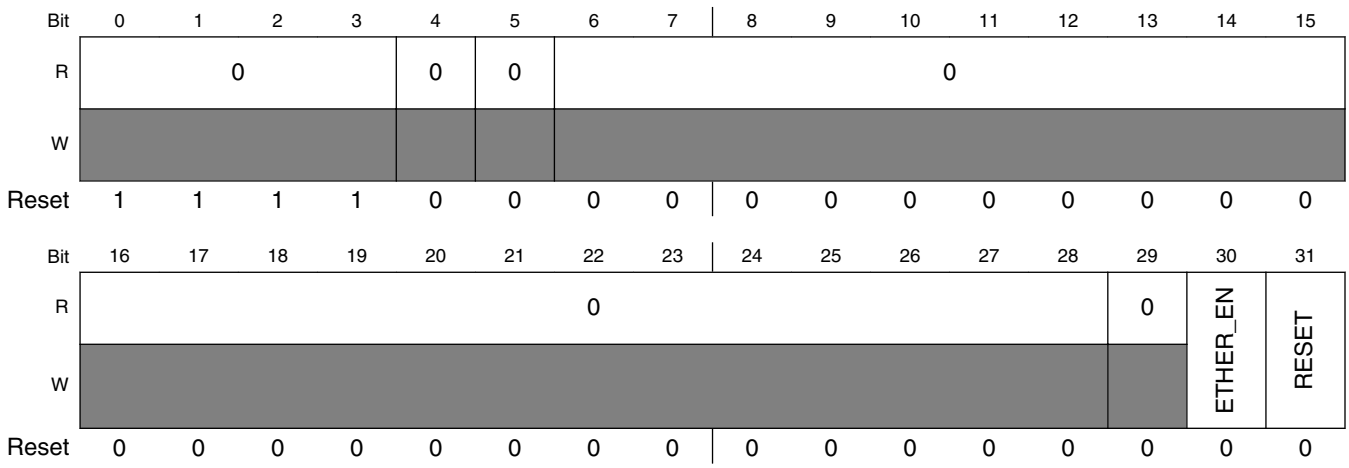
FEC_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.5 Ethernet Control Register (FEC_ECR)

ECR is a read/write user register, though hardware may alter fields in this register as well. The ECR enables/disables the FEC.

Address: 0h base + 24h offset = 24h



FEC_ECR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN is cleared. • An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared.
31 RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ECR[ETHER_EN] is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately eight internal bus clock cycles after this bit is set.

53.4.6 MII Management Frame Register (FEC_MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR causes a management frame to be sourced unless the MSCR is programmed to 0. If MSCR is cleared while MMFR is written and then MSCR is written with a non-zero value, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

The MMFR does not reset to any predefined value.

To perform a read or write operation on the MII Management Interface, write the MMFR. To generate a valid read or write management frame, ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), write {01 01 PHYAD REGAD 10 DATA} to the MMFR. Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the write management frame operation completes, the MII interrupt is generated. At this time, contents of the MMFR match the original value written.

To generate an MII management interface read frame (read a PHY register), write {01 10 PHYAD REGAD 10 XXXX} to the MMFR (the content of the DATA field is unimportant). Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the read management frame operation completes, the MII interrupt is generated. At this time, the contents of the MMFR register match the original value written except for the DATA field whose contents are replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST		OP		PA					RA					TA		DATA															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

FEC_MMFR field descriptions

Field	Description
0–1 ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
2–3 OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
4–8 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
9–13 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
14–15 TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

53.4.7 MII Speed Control Register (FEC_MSCR)

The MSCR:

- Provides control of the MII clock (FEC_MDC pin) frequency
- Allows a preamble drop on the MII management frame

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be set to 0 to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED changes during operation (change takes effect following a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0000_0005h results in an FEC_MDC as stated the equation below.

$$25\,\mathrm{MHz} \times \frac{1}{5 \times 2} = 2.5\,\mathrm{MHz}$$

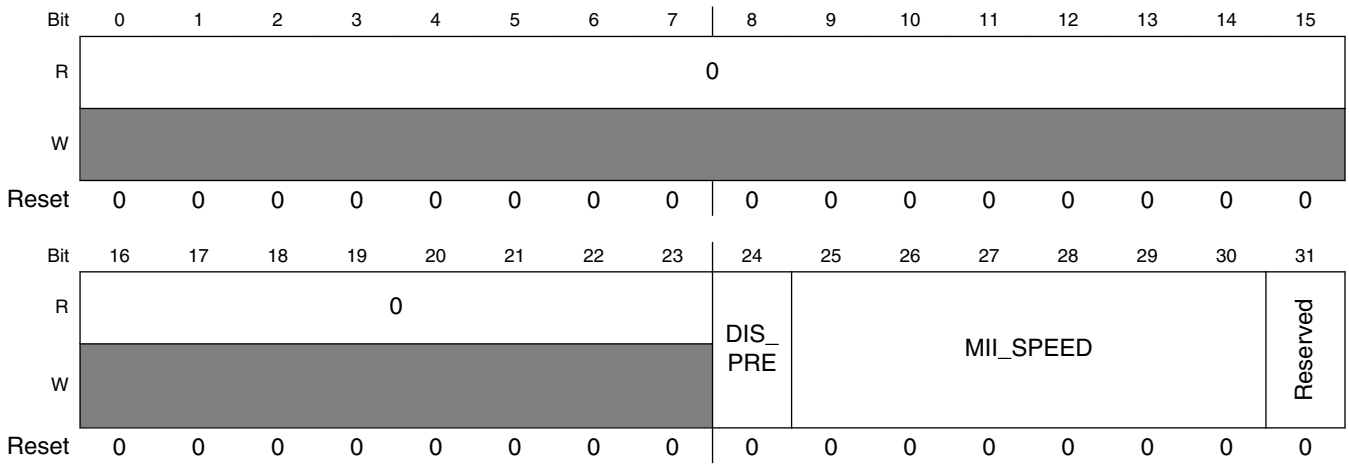
Equation 30. FEC_MDC example

Table 53-4 shows optimum values for MII_SPEED as a function of internal bus clock frequency.

Table 53-4. Programming examples for MSCR

Internal FEC clock frequency	MSCR[MII_SPEED]	FEC_MDC frequency
25 MHz	5h	2.50 MHz
33 MHz	7h	2.36 MHz
40 MHz	8h	2.50 MHz
50 MHz	Ah	2.50 MHz
66 MHz	Eh	2.36 MHz
80 MHz	10h	2.50 MHz
83 MHz	10h	2.59 MHz

Address: 0h base + 44h offset = 44h



FEC_MSCR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 DIS_PRE	Setting this bit causes the preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
25–30 MII_SPEED	Controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED x 2) of the internal bus frequency.
31 Reserved	This field is reserved. Reserved

53.4.8 MIB Control Register (FEC_MIBC)

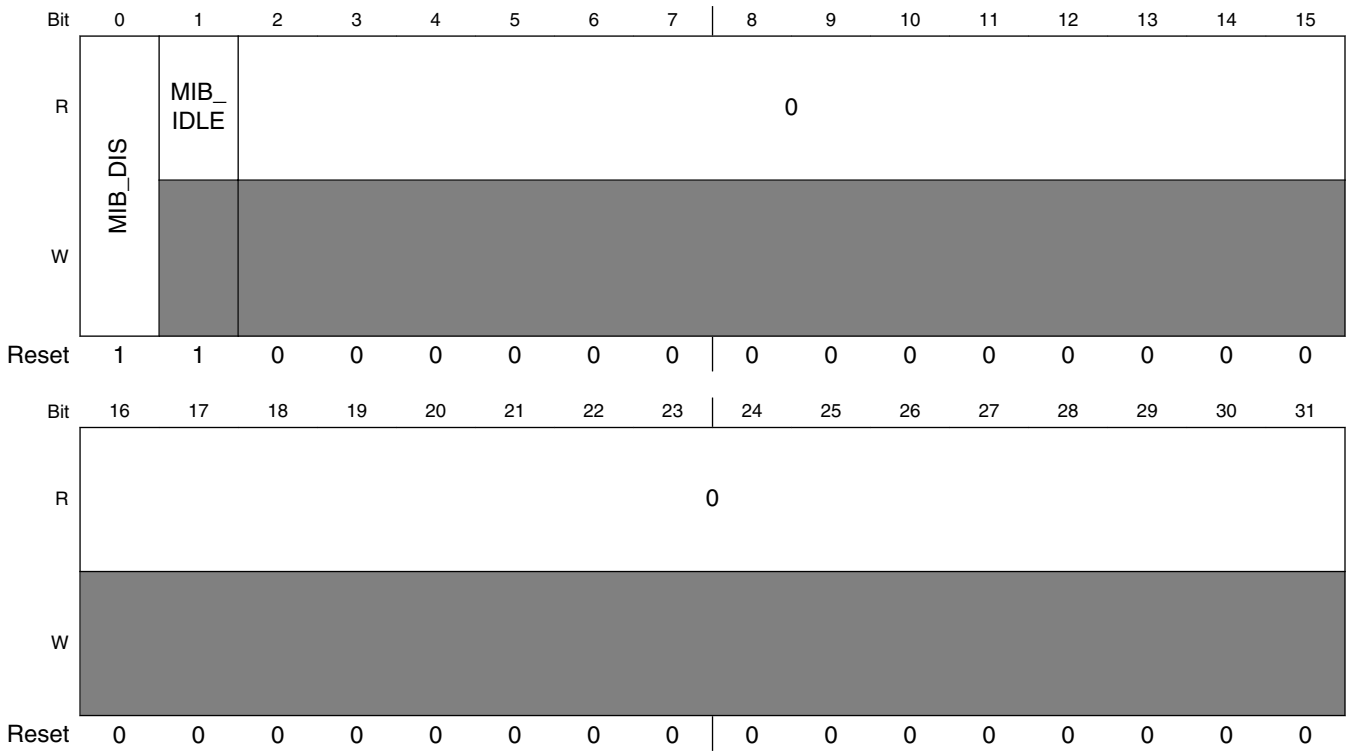
The MIBC is a read/write register controlling and observing the state of the MIB block. User software accesses this register if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM:

1. Disable the MIB block

2. Clear all the MIB RAM locations
3. Enable the MIB block

The MIB_DIS bit is reset to 1. See the memory map for the locations of the MIB counters.

Address: 0h base + 64h offset = 64h



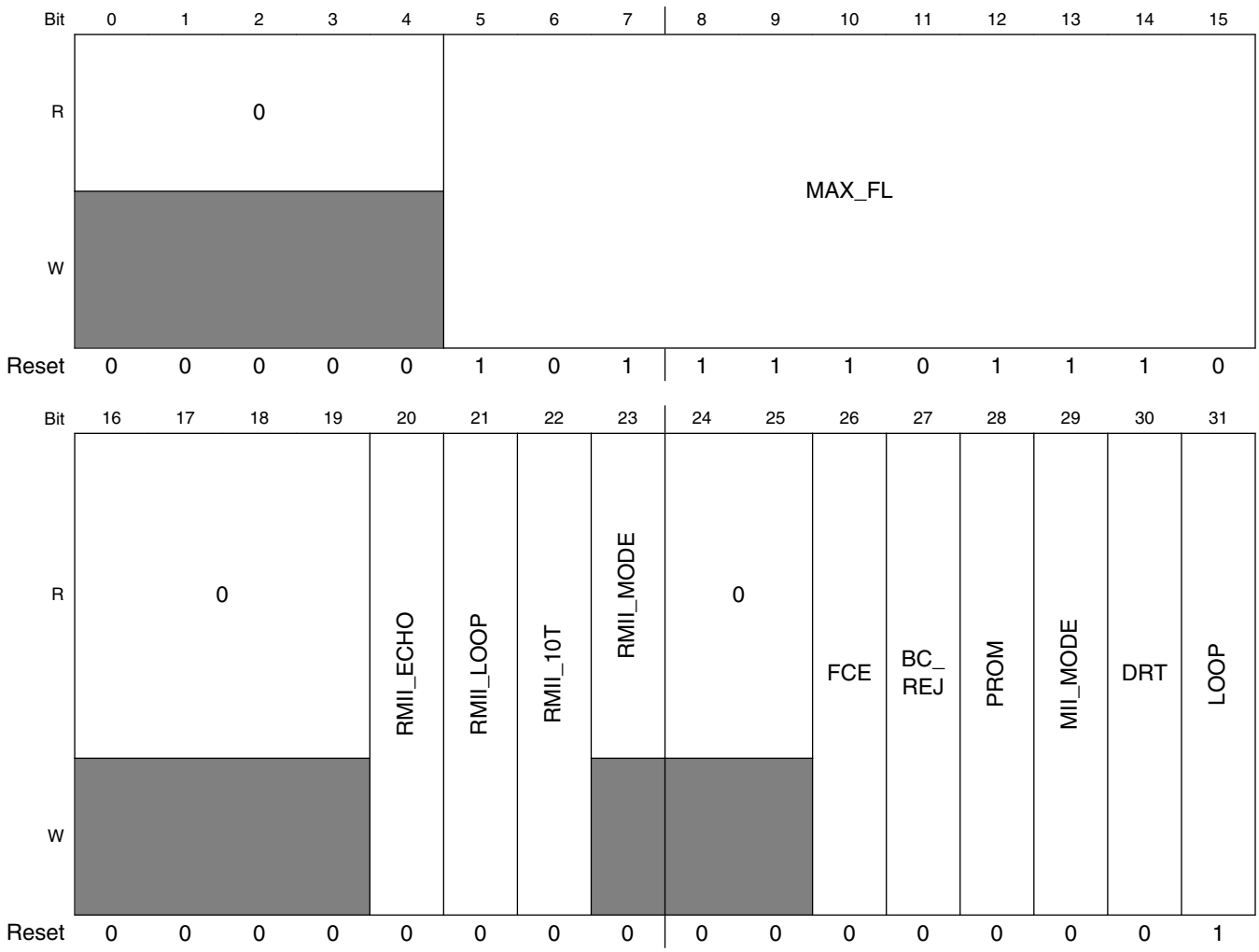
FEC_MIBC field descriptions

Field	Description
0 MIB_DIS	If MIB_DIS = 1, the MIB logic halts and not update any MIB counters.
1 MIB_IDLE	If MIB_IDLE = 1, the MIB block is not currently updating any MIB counters.
2–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.9 Receive Control Register (FEC_RCR)

RCR controls the operational mode of the receive block and must be written only when ECR[ETHER_EN] is cleared (initialization time).

Address: 0h base + 84h offset = 84h



FEC_RCR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 MAX_FL	Maximum frame length. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor.

Table continues on the next page...

FEC_RCR field descriptions (continued)

Field	Description
	The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 RMII_ECHO	RMII Echo. Enables RMII echo mode. RMII 2-bit receive data is processed through the RMII receive logic to the FEC and also routes through the RMII's transmit logic to become RMII 2-bit transmit data out. 0 Normal operation. 1 RMII echo mode enabled. **NOTE:** When RMII_ECHO is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_LOOP = 0 and LOOP = 0.
21 RMII_LOOP	RMII loopback. Enables RMII loopback mode. Causes the MII transmit outputs from the Ethernet controller to loop back to the Ethernet controllers's MII receive inputs through the RMII transmit/receive logic. 0 Normal operation. 1 RMII loopback mode enabled. **NOTE:** When RMII_LOOP is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_ECHO = 0, LOOP = 0, and TCR *n* [FDEN] = 1.
22 RMII_10T	RMII 10-Base T. Enables 10Mbps mode of the RMII. Determines the clock frequency of the clock source to the FEC logic to support 10/100Mbps operations. 0 100 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is sent to the RMII, while a divided-by-2 version (25 MHz) is sent to the FEC. 1 10 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is divided by 10 (5 MHz) and sent to the RMII, while a divided-by-20 version (2.5 MHz) is sent to the FEC.
23 RMII_MODE	RMII Mode. Indicates if the FEC is in RMII or MII/7-wire mode. See the chip configuration chapter for more details. **NOTE:** Interface selection is chip-specific; see the chapter that describes how modules are configured and connected. 0 FEC is in true MII mode. 1 FEC is in RMII mode.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equal to FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
28 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
29 MII_MODE	Media independent interface mode. Selects the external interface mode for transmit and receive blocks. **NOTE:** Interface selection is chip-specific; see the chapter that describes how modules are configured and connected. 0 7-wire mode (used only for serial 10 Mbps) 1 MII or RMII mode as indicated by the RMII_MODE bit

Table continues on the next page...

FEC_RCR field descriptions (continued)

Field	Description
30 DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
31 LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

53.4.10 Transmit Control Register (FEC_TCR)

The TCR configures the transmit block.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TCR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
29 FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This field must be modified only when ECR[ETHER_EN] = 0.
30 HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This field must be modified only when ECR[ETHER_EN] = 0.
31 GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this situation, program ECR[ETHER_EN] = 0 following the GRA interrupt.

53.4.11 Physical Address Low Register (FEC_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames.

This register is not reset and you must initialize it.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_PALR field descriptions

Field	Description
0–31 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

53.4.12 Physical Address High Register and Type Field (FEC_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (8808h) for transmission of PAUSE frames. The upper 16 bits of this register are not reset and you must initialize them.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR2																TYPE															
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

* Notes:
- u = Unaffected by reset.

FEC_PAUR field descriptions

Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type field in PAUSE frames. These 16 read-only bits are a constant value of 8808h.

53.4.13 Opcode/Pause Duration (FEC_OPD)

The OPD contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0001h. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field.

The lower 16 bits of this register are not reset and you must initialize them.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode field used in PAUSE frames. These read-only bits are a constant, 0001h.
16–31 PAUSE_DUR	Pause Duration field used in PAUSE frames.

53.4.14 Descriptor Individual Upper Address Register (FEC_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_IAUR field descriptions

Field	Description
0–31 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

53.4.15 Descriptor Individual Lower Address Register (FEC_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR2																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_IALR field descriptions

Field	Description
0–31 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

53.4.16 Descriptor Group Upper Address Register (FEC_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_GAUR field descriptions

Field	Description
0–31 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

53.4.17 Descriptor Group Lower Address Register (FEC_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_GALR field descriptions

Field	Description
0–31 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

53.4.18 Transmit FIFO Watermark (FEC_TFWR)

The TFWR controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														TFWR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TFWR field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written

Table continues on the next page...

FEC_TFWR field descriptions (continued)

Field	Description
	01 64 bytes written 10 128 bytes written 11 192 bytes written

53.4.19 FIFO Receive Bound Register (FEC_FRBR)

FRBR indicates the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

Address: 0h base + 14Ch offset = 14Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					1	R_BOUND								0	
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

FEC_FRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–29 R_BOUND	Highest valid FIFO RAM address.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.20 FIFO Receive Start Register (FEC_FRSR)

FRSR indicates the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

Hardware initializes the FRSR at reset. FRSR only needs to be written to change the default value.

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					1	R_FSTART								0	
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

FEC_FRSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–29 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 48h or greater.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.21 Receive Descriptor Ring Start Register (FEC_ERDSR)

ERDSR points to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START																														0	
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_ERDSR field descriptions

Field	Description
0–29 R_DES_START	Pointer to start of receive buffer descriptor queue.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.22 Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)

ETSDR provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	X_DES_START																														0	
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_ETDSR field descriptions

Field	Description
0–29 X_DES_START	Pointer to start of transmit buffer descriptor queue.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.23 Receive Buffer Size Register (FEC_EMRBR)

The EMRBR dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. To properly align the buffer, EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low.

To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR is undefined at reset and must be initialized by the user.

Address: 0h base + 188h offset = 188h

Bit	0–20	21–27	28–31
R	0	R_BUF_SIZE	0
W			
Reset	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_EMRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Maximum size of receive buffer size in bytes. To minimize bus utilization (descriptor fetches), set this field to 256 bytes (10h) or larger. 10h: 256 + 15 bytes (minimum size recommended) 11h: 272 + 15 bytes ... 7Fh: 2032 + 15 bytes. The FEC writes up to 2047 bytes in the receive buffer. If data larger than 2047 is received, the FEC truncates it and shows 7FFh in the receive descriptor
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.24 Count of frames not counted correctly (FEC_RMON_T_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 200h offset = 200h

Bit	0–31
R W	RMON_T_DROP
Reset	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_DROP field descriptions

Field	Description
0–31 RMON_T_DROP	Count of frames not counted correctly

53.4.25 RMON Tx packet count (FEC_RMON_T_PACKETS)

See Using the RMON and IEEE registers.

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_PACKETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_PACKETS field descriptions

Field	Description
0–31 RMON_T_ PACKETS	RMON Tx packet count

53.4.26 RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 208h offset = 208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_BC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_BC_PKT field descriptions

Field	Description
0–31 RMON_T_BC_ PKT	RMON Tx broadcast packets

53.4.27 RMON Tx multicast packets (FEC_RMON_T_MC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 20Ch offset = 20Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_MC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_MC_PKT field descriptions

Field	Description
0–31 RMON_T_MC_ PKT	RMON Tx multicast packets

53.4.28 RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN)

See Using the RMON and IEEE registers.

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_CRC_ALIGN																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_CRC_ALIGN field descriptions

Field	Description
0–31 RMON_T_CRC_ ALIGN	RMON Tx packets with CRC/align error

53.4.29 RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_UNDERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_UNDERSIZE field descriptions

Field	Description
0–31 RMON_T_ UNDERSIZE	RMON Tx packets < 64 bytes, good CRC

53.4.30 RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 218h offset = 218h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_OVERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_OVERSIZE field descriptions

Field	Description
0–31 RMON_T_ OVERSIZE	RMON Tx packets > MAX_FL bytes, good CRC

53.4.31 RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG)

See Using the RMON and IEEE registers.

Address: 0h base + 21Ch offset = 21Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_FRAG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_FRAG field descriptions

Field	Description
0–31 RMON_T_FRAG	RMON Tx packets < 64 bytes, bad CRC

53.4.32 RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB)

See Using the RMON and IEEE registers.

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_JAB																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_JAB field descriptions

Field	Description
0–31 RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad CRC

53.4.33 RMON Tx collision count (FEC_RMON_T_COL)

See Using the RMON and IEEE registers.

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_COL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_COL field descriptions

Field	Description
0–31 RMON_T_COL	RMON Tx collision count

53.4.34 RMON Tx 64 byte packets (FEC_RMON_T_P64)

See Using the RMON and IEEE registers.

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P64																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P64 field descriptions

Field	Description
0–31 RMON_T_P64	RMON Tx 64 byte packets

53.4.35 RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127)

See Using the RMON and IEEE registers.

Address: 0h base + 22Ch offset = 22Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P65TO127																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P65TO127 field descriptions

Field	Description
0–31 RMON_T_ P65TO127	RMON Tx 65 to 127 byte packets

53.4.36 RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255)

See Using the RMON and IEEE registers.

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P128TO255																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P128TO255 field descriptions

Field	Description
0–31 RMON_T_ P128TO255	RMON Tx 128 to 255 byte packets

53.4.37 RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511)

See Using the RMON and IEEE registers.

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P256TO511																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P256TO511 field descriptions

Field	Description
0–31 RMON_T_ P256TO511	RMON Tx 256 to 511 byte packets

53.4.38 RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023)

See Using the RMON and IEEE registers.

Address: 0h base + 238h offset = 238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P512TO1023																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P512TO1023 field descriptions

Field	Description
0–31 RMON_T_ P512TO1023	RMON Tx 512 to 1023 byte packets

53.4.39 RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047)

See Using the RMON and IEEE registers.

Address: 0h base + 23Ch offset = 23Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P1024TO2047																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P1024TO2047 field descriptions

Field	Description
0–31 RMON_T_ P1024TO2047	RMON Tx 1024 to 2047 byte packets

53.4.40 RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048)

See Using the RMON and IEEE registers.

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P_GTE2048																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P_GTE2048 field descriptions

Field	Description
0–31 RMON_T_P_ GTE2048	RMON Tx packets with > 2048 bytes

53.4.41 RMON Tx Octets (FEC_RMON_T_OCTETS)

See Using the RMON and IEEE registers.

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_OCTETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_OCTETS field descriptions

Field	Description
0–31 RMON_T_ OCTETS	RMON Tx Octets

53.4.42 Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 248h offset = 248h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_DROP field descriptions

Field	Description
0–31 IEEE_T_DROP	Count of transmitted frames not counted correctly

53.4.43 Frames transmitted OK (FEC_IEEE_T_FRAME_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 24Ch offset = 24Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_FRAME_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_FRAME_OK field descriptions

Field	Description
0–31 IEEE_T_ FRAME_OK	Frames transmitted OK

53.4.44 Frames transmitted with single collision (FEC_IEEE_T_1COL)

See Using the RMON and IEEE registers.

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_1COL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

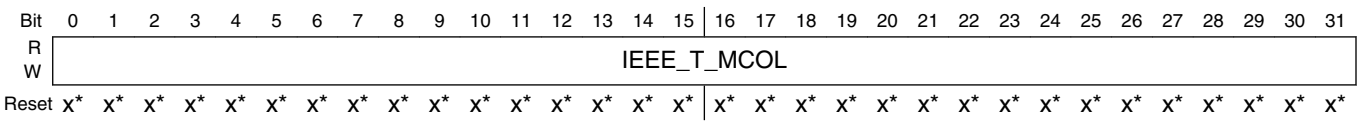
FEC_IEEE_T_1COL field descriptions

Field	Description
0–31 IEEE_T_1COL	Frames transmitted with single collision

53.4.45 Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL)

See Using the RMON and IEEE registers.

Address: 0h base + 254h offset = 254h



* Notes:
- x = Undefined at reset.

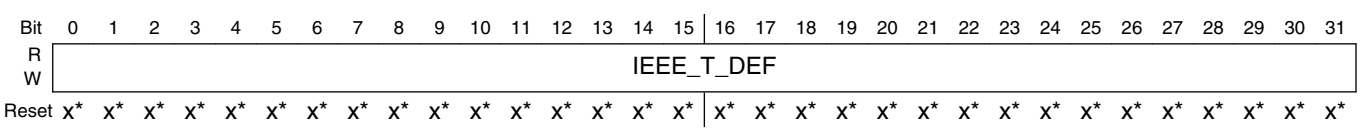
FEC_IEEE_T_MCOL field descriptions

Field	Description
0–31 IEEE_T_MCOL	Frames transmitted with multiple collisions

53.4.46 Frames transmitted after deferral delay (FEC_IEEE_T_DEF)

See Using the RMON and IEEE registers.

Address: 0h base + 258h offset = 258h



* Notes:
- x = Undefined at reset.

FEC_IEEE_T_DEF field descriptions

Field	Description
0–31 IEEE_T_DEF	Frames transmitted after deferral delay

53.4.47 Frames transmitted with late collision (FEC_IEEE_T_LCOL)

See Using the RMON and IEEE registers.

Address: 0h base + 25Ch offset = 25Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_LCOL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_LCOL field descriptions

Field	Description
0–31 IEEE_T_LCOL	Frames transmitted with late collision

53.4.48 Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL)

See Using the RMON and IEEE registers.

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_EXCOL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_EXCOL field descriptions

Field	Description
0–31 IEEE_T_EXCOL	Frames transmitted with excessive collisions

53.4.49 Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR)

See Using the RMON and IEEE registers.

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_MACERR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_MACERR field descriptions

Field	Description
0–31 IEEE_T_ MACERR	Frames transmitted with Tx FIFO underrun

53.4.50 Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR)

See Using the RMON and IEEE registers.

Address: 0h base + 268h offset = 268h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_CSERR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_CSERR field descriptions

Field	Description
0–31 IEEE_T_CSERR	Frames transmitted with carrier sense error

53.4.51 Frames transmitted with SQE error (FEC_IEEE_T_SQE)

See Using the RMON and IEEE registers.

Address: 0h base + 26Ch offset = 26Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_SQE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_SQE field descriptions

Field	Description
0–31 IEEE_T_SQE	Frames transmitted with SQE error

53.4.52 Flow control pause frames transmitted (FEC_IEEE_T_FDXFC)

See Using the RMON and IEEE registers.

Address: 0h base + 270h offset = 270h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_FDXFC																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_FDXFC field descriptions

Field	Description
0–31 IEEE_T_FDXFC	Flow control pause frames transmitted

53.4.53 Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 274h offset = 274h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_OCTETS_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 IEEE_T_ OCTETS_OK	Octet count for frames transmitted without error

53.4.54 Count of received frames not counted correctly (FEC_RMON_R_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 280h offset = 280h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_DROP field descriptions

Field	Description
0–31 RMON_R_DROP	Count of received frames not counted correctly

53.4.55 RMON Rx packet count (FEC_RMON_R_PACKETS)

See Using the RMON and IEEE registers.

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_PACKETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_PACKETS field descriptions

Field	Description
0–31 RMON_R_ PACKETS	RMON Rx packet count

53.4.56 RMON Rx broadcast packets (FEC_RMON_R_BC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 288h offset = 288h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_BC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_BC_PKT field descriptions

Field	Description
0–31 RMON_R_BC_ PKT	RMON Rx broadcast packets

53.4.57 RMON Rx multicast packets (FEC_RMON_R_MC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 28Ch offset = 28Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_MC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_MC_PKT field descriptions

Field	Description
0–31 RMON_R_MC_PKT	RMON Rx multicast packets

53.4.58 RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN)

See Using the RMON and IEEE registers.

Address: 0h base + 290h offset = 290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_CRC_ALIGN																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–31 RMON_R_CRC_ALIGN	RMON Rx packets with CRC/Align error

53.4.59 RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 294h offset = 294h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_UNDERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_UNDERSIZE field descriptions

Field	Description
0–31 RMON_R_ UNDERSIZE	RMON Rx packets < 64 bytes, good CRC

53.4.60 RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 298h offset = 298h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_OVERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_OVERSIZE field descriptions

Field	Description
0–31 RMON_R_ OVERSIZE	RMON Rx packets > MAX_FL bytes, good CRC

53.4.61 RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)

See Using the RMON and IEEE registers.

Address: 0h base + 29Ch offset = 29Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_FRAG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_FRAG field descriptions

Field	Description
0–31 RMON_R_FRAG	RMON Rx packets < 64 bytes, bad CRC

53.4.62 RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)

See Using the RMON and IEEE registers.

Address: 0h base + 2A0h offset = 2A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_JAB																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_JAB field descriptions

Field	Description
0–31 RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad CRC

53.4.63 Reserved (FEC_RMON_R_RESVD_0)

See Using the RMON and IEEE registers.

Address: 0h base + 2A4h offset = 2A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_RESVD_0																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_RESVD_0 field descriptions

Field	Description
0–31 RMON_R_ RESVD_0	Reserved

53.4.64 RMON Rx 64 byte packets (FEC_RMON_R_P64)

See Using the RMON and IEEE registers.

Address: 0h base + 2A8h offset = 2A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P64																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P64 field descriptions

Field	Description
0–31 RMON_R_P64	RMON Rx 64 byte packets

53.4.65 RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)

See Using the RMON and IEEE registers.

Address: 0h base + 2ACh offset = 2ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P65TO127																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P65TO127 field descriptions

Field	Description
0–31 RMON_R_ P65TO127	RMON Rx 65 to 127 byte packets

53.4.66 RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)

See Using the RMON and IEEE registers.

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P128TO255																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P128TO255 field descriptions

Field	Description
0–31 RMON_R_ P128TO255	RMON Rx 128 to 255 byte packets

53.4.67 RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511)

See Using the RMON and IEEE registers.

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P256TO511																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P256TO511 field descriptions

Field	Description
0–31 RMON_R_ P256TO511	RMON Rx 256 to 511 byte packets

53.4.68 RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023)

See Using the RMON and IEEE registers.

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P512TO1023																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P512TO1023 field descriptions

Field	Description
0–31 RMON_R_ P512TO1023	RMON Rx 512 to 1023 byte packets

53.4.69 RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)

See Using the RMON and IEEE registers.

Address: 0h base + 2BCh offset = 2BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P1024TO2047																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P1024TO2047 field descriptions

Field	Description
0–31 RMON_R_ P1024TO2047	RMON Rx 1024 to 2047 byte packets

53.4.70 RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)

See Using the RMON and IEEE registers.

Address: 0h base + 2C0h offset = 2C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P_GTE2048																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P_GTE2048 field descriptions

Field	Description
0–31 RMON_R_P_ GTE2048	RMON Rx packets with > 2048 bytes

53.4.71 RMON Rx octets (FEC_RMON_R_OCTETS)

See Using the RMON and IEEE registers.

Address: 0h base + 2C4h offset = 2C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_OCTETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_OCTETS field descriptions

Field	Description
0–31 RMON_R_ OCTETS	RMON Rx octets

53.4.72 Count of received frames not counted correctly (FEC_IEEE_R_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 2C8h offset = 2C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_DROP field descriptions

Field	Description
0–31 IEEE_R_DROP	Count of received frames not counted correctly

53.4.73 Frames received OK (FEC_IEEE_R_FRAME_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 2CCh offset = 2CCh

Bit	0–31
R W	IEEE_R_FRAME_OK
Reset	x* … x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_FRAME_OK field descriptions

Field	Description
0–31 IEEE_R_ FRAME_OK	Frames received OK

53.4.74 Frames received with CRC error (FEC_IEEE_R_CRC)

See Using the RMON and IEEE registers.

Address: 0h base + 2D0h offset = 2D0h

Bit	0–31
R W	IEEE_R_CRC
Reset	x* … x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_CRC field descriptions

Field	Description
0–31 IEEE_R_CRC	Frames received with CRC error

53.4.75 Frames received with alignment error (FEC_IEEE_R_ALIGN)

See Using the RMON and IEEE registers.

Address: 0h base + 2D4h offset = 2D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_ALIGN																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_ALIGN field descriptions

Field	Description
0–31 IEEE_R_ALIGN	Frames received with alignment error

53.4.76 Receive FIFO overflow count (FEC_IEEE_R_MACERR)

See Using the RMON and IEEE registers.

Address: 0h base + 2D8h offset = 2D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_MACERR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_MACERR field descriptions

Field	Description
0–31 IEEE_R_ MACERR	Receive FIFO overflow count

53.4.77 Flow control pause frames received (FEC_IEEE_R_FDXFC)

See Using the RMON and IEEE registers.

Address: 0h base + 2DCh offset = 2DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_FDXFC																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_FDXFC field descriptions

Field	Description
0–31 IEEE_R_FDXFC	Flow control pause frames received

53.4.78 Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_OCTETS_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 IEEE_R_ OCTETS_OK	Octet count for frames received without error

53.4.79 Using the RMON and IEEE registers

In order to use the RMON and IEEE registers to monitor FEC performance, you must prepare them using the sequence below:

1. Write zeros to the registers.
2. Enable the FEC.

Until you complete this sequence, the registers are read/write, and their reset value is undefined. After you complete this sequence, the registers become read-only, and you may read them as necessary.

53.5 Functional description

This section describes the operation of the FEC, beginning with the description of two different types of MII frames, the buffer descriptors, the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

53.5.1 MII data frame

Ethernet/802.3 data frames transmitted across the MII have the format specified in Figure 53-2 and Table 53-5.

<inter-frame> <preamble> <sfd> <data> <efd>

Figure 53-2. MII data frame format

Table 53-5. MII data frame components

Component	Description
<inter-frame>	Inter-frame period. This is an unspecified amount of time during which no data activity occurs on the MII. The de-assertion of RX_DV and TX_EN indicate the absence of data activity.
<preamble>	Preamble that begins a frame. It has the following value: 10101010 10101010 10101010 10101010 10101010 10101010 10101010 The leftmost 1 represents the LSB of the byte.
<sfd>	Start of a frame. It has the value 10101011.
<data>	Data portion of the frame. It consists of N octets which corresponds to 2N nibbles being transmitted. The order of each nibble is defined in Figure 53-3.
<efd>	End-of-frame delimiter.

Table 53-5. MII data frame components

Component	Description
	For TXD data, it is indicated by the de-assertion of the TX_EN signal. For RXD data, it is indicated by the de-assertion of the RX_DV signal.

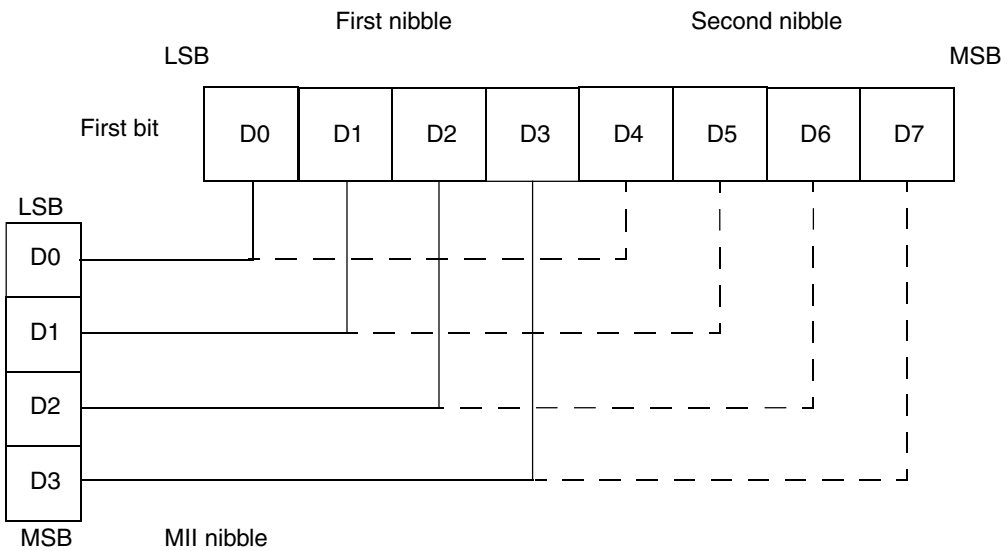


Figure 53-3. MII nibble/octet to octet/nibble mapping

53.5.2 MII management frame structure

A transceiver management frame being transmitted on the MII management interface uses the MDIO and MDC pins. A transaction or frame on this serial interface has the format specified in Figure 53-4 and Table 53-6.

<preamble> <st> <op> <phyad> <regad> <ta> <data> <idle>

Figure 53-4. MII management frame format

Table 53-6. MII management frame components

Component	Description
<preamble>	Optional preamble, consisting of a sequence of 32 continuous logic 1's.
<st>	Start of frame, consisting of a 01 pattern.
<op>	Operation code. 01 Write operation 10 Read instruction
<phyad>	Five-bit field that allows up to 32 PHYs to be addressed. The first address bit transmitted is the MSB of the address.

Table continues on the next page...

Table 53-6. MII management frame components (continued)

Component	Description
<regad>	Five-bit field that allows for 32 registers to be addressed within each PHY. The first register bit transmitted is the MSB of the address.
<ta>	Two-bit field that provides spacing between the register address field and the data field to avoid contention on the MDIO signal during a read operation.
<data>	16-bit data field. Bit 15 is the first data bit transmitted and received.
<idle>	Idle condition (MDIO is in the high-impedance state).

The MII management register set located in the PHY may consist of a basic register set and an extended register set as defined in Table 53-7.

Table 53-7. MII management register set

Register address	Register name	Register set
0	Control	Basic
1	Status	
2-3	PHY identifier	Extended
4	Auto-negotiation advertisement	
5	AN link partner ability	
6	AN expansion	
7	AN next page transmit	
8-15	Reserved	
16-31	Vendor-specific	

53.5.3 Buffer descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

53.5.3.1 Driver/DMA operation with buffer descriptors

The data for the FEC frames resides in one or more memory buffers external to the FEC. Associated with each buffer is a buffer descriptor (BD), which contains a starting address (32-bit aligned pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read by the FEC DMA engine.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit produces the buffer. Software writing to TDAR or RDAR tells the FEC that a buffer is placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the DMA engine writes the buffer descriptor status bits, hardware clears RxBD[E] or TxBD[R] to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers are consumed or may rely on the buffer/frame interrupts. The driver may process these buffers, and they can return to the free list.

The ECR[ETHER_EN] bit operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before ECR[ETHER_EN] is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The wrap (W) bit defines the last buffer descriptor in each ring. When W is set, the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they be 128-bit aligned.

53.5.3.1.1 Driver/DMA operation with transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, and Ethernet/IEEE 802.3 header in a fourth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type field(s)), so the driver must provide this in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. TxBD[TC], which must be set by the driver, determines whether the MAC or driver appends the CRC.

The driver (TxBD software producer) should set up Tx BDs so a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC) and then the TxBD[R] bit should be set in reverse order (third, second, then first BD) to ensure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to TDAR. When this register is written to (data value is not significant), the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared. At this point, the FEC polls this BD one more time. If the R bit is cleared the second time, RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

53.5.3.1.2 Driver/DMA operation with receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L is set), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not only the length of the last buffer.

For simplicity, the driver may assign a large enough default receive buffer length to contain an entire frame, keeping in mind that a malfunction on the network or out-of-spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software never sees a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received, the FEC fills receive buffers and updates the associated BDs, then reads the next BD in the receive

descriptor ring. If the FEC reads a receive BD and finds the E bit cleared, it polls this BD once more. If RxBD[E] is clear a second time, FEC stops reading receive BDs until the driver writes to RDAR.

53.5.3.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first longword and the pointer in the second longword. When the buffer has been DMA'd, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

Table 53-8. Receive Buffer Descriptor (RxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset+0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset+2	Data Length															
Offset+4	Rx Data Buffer Pointer - A[31:16]															
Offset+6	Rx Data Buffer Pointer - A[15:0]															

Table 53-9. Receive Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	E	Empty. Written by the FEC (=0) and user (=1). 0: The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1: The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0: The next buffer descriptor is found in the consecutive location 1: The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by the FEC. 0: The buffer is not the last in a frame. 1: The buffer is the last in a frame.
Offset + 0	M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0: The frame was received because of an address recognition hit.

Table continues on the next page...

Table 53-9. Receive Buffer Descriptor field descriptions (continued)

Word	Field	Description
		1: The frame was received because of promiscuous mode.
Offset + 0	BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 0	MC	Set if the DA is multicast and not BC.
Offset + 0	LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set, the CR bit is not set.
Offset + 0	CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the FEC once as the BD is closed.
0ffset + 4	A[31:16]	RX data buffer pointer, bits [31:16][1]
Offset + 6	A[15:0]	RX data buffer pointer, bits [15:0]

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

Note

When the software driver sets an E bit in one or more receive descriptors, the driver should follow with a write to RDAR.

53.5.3.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (TxBD[R]) when DMA of the buffer is complete. In the TxBD, the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword and the buffer pointer in the second longword.

The FEC clears the R bit when the buffer is transferred. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block.

Table 53-10. Transmit Buffer Descriptor (TxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset+0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset+2	Data Length															
Offset+4	Tx Data Buffer Pointer - A[31:16]															
Offset+6	Tx Data Buffer Pointer - A[15:0]															

Table 53-11. Transmit Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	R	Ready. Written by the FEC and you. 0: The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1: The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this bit is set.
Offset + 0	TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0: The next buffer descriptor is found in the consecutive location 1: The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by user. 0: The buffer is not the last in the transmit frame 1: The buffer is the last in the transmit frame
Offset + 0	TC	Transmit CRC. Written by user (only valid if L is set). 0: End transmission immediately after the last data byte 1: Transmit the CRC sequence after the last data byte
Offset + 0	ABC	Append bad CRC. Written by user (only valid if L is set). 0: No effect 1: Transmit the CRC sequence inverted after the last data byte (regardless of TC value)
Offset + 2	Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC.
Offset + 4	A[31:16]	Tx data buffer pointer, bits [31:16] [1]
Offset + 6	A[15:0]	Tx data buffer pointer, bits [15:0]

1. The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

Note

After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame is setting the R bit in the first BD for the frame. The driver must follow that with a write to TDAR that triggers the FEC to poll the next BD in the ring.

53.5.4 Initialization sequence

This section describes which registers are reset due to hardware reset, which registers are reset by the FEC RISC, and what locations you must initialize prior to enabling the FEC.

53.5.4.1 Hardware Controlled Initialization

In the FEC, hardware resets registers and control logic that generate interrupts. A hardware reset negates output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared (which is accomplished by a hard reset or software to halt operation). By clearing ECR[ETHER_EN], configuration control registers such as the TCR and RCR are not reset, but the entire data path is reset.

Table 53-12. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

53.5.5 User initialization (Prior to Setting ECR[ETHER_EN])

You need to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The sequence is not important.

The following registers require initialization:

- EIMR
- EIR (clear by writing FFFF_FFFFh)
- TFWR (optional)
- IALR / IAUR
- GAUR / GALR
- PALR / PAUR (only needed for full duplex flow control)
- OPD (only needed for full duplex flow control)
- RCR
- TCR
- MSCR (optional)
- MIB_RAM (clear)
- FRSR (optional)
- EMRBR
- ERDSR
- ETDSR
- Transmit Descriptor ring (empty)
- Receive Descriptor ring (empty)

53.5.6 Microcontroller initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, hardware is ready for operation.

The sequence is as follows:

1. Initialize the backoff random number seed.
2. Activate receiver.
3. Activate transmitter.
4. Clear transmit FIFO.

5. Clear receive FIFO.
6. Initialize the transmit ring pointer.
7. Initialize the receive ring pointer.
8. Initialize FIFO count registers.

53.5.7 User initialization (after setting ECR[ETHER_EN])

After setting ECR[ETHER_EN], you can set up the buffer/frame descriptors and write to TDAR and RDAR. See Buffer descriptors, for more details.

53.5.8 Network interface options

The FEC supports an MII and reduced MII interface for 10/100 Mbps Ethernet, as well as a 7-wire serial interface for 10 Mbps Ethernet. The RCR[MII_MODE] and RCR [RMII_MODE] bits select the interface mode. In MII mode (RCR[MII_MODE]=1 and RCR [RMII_MODE]=0), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. The following table shows these signals.

Table 53-13. MII mode

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[3:0]
Transmit Error	FEC_TXER
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RXER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

In RMII mode (RCR[MII_MODE]=1 and RCR[RMII_MODE]=1), the EMAC supports 8 external signals. These signals are shown in the table below.

Table 53-14. RMII mode configuration

Signal description	EMAC pin
Reference Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[1:0]
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[1:0]
Receive Error	FEC_RXER

The 7-wire serial mode interface (RCR[MII_MODE]=1 and RCR[RMII_MODE]=0 or 1) is generally referred to as AMD mode. The following table shows the 7-wire mode connections to the external transceiver.

Table 53-15. 7-Wire mode configuration

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[0]

53.5.9 FEC frame transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set and data appears in the transmit FIFO, the Ethernet MAC can transmit onto the network. The Ethernet controller transmits bytes least significant bit (lsb) first.

When the transmit FIFO fills to the watermark (defined by TFWR), MAC transmit logic asserts FEC_TXEN and starts transmitting the (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See "Transmission errors" for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data is transmitted, FCS (frame check sequence) or 32-bit cyclic redundancy check (CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Transmit logic automatically pads short frames (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set).

Settings in the EIMR determine interrupts generated to the buffer (TXB) and frame (TXF).

The transmit error interrupts are HBERR, BABT, LC, RL, and UN. If the transmit frame length exceeds MAX_FL bytes, BABT interrupt is asserted. However, the entire frame is transmitted (no truncation).

To pause transmission, set TCR[GTS] (graceful transmit stop). The FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

53.5.9.1 Duplicate frame transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, *n.* The minimum number of TxBDs is then (Tx FIFO Size ÷ (*n*+4)) rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

53.5.10 FEC frame reception

The FEC receiver works with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking. The Ethernet controller receives serial data lsb first.

When the driver enables the FEC receiver by setting ECR[ETHER_EN], it immediately starts processing receive frames. When FEC_RXDV is asserted, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the receiver processes the frame. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RXDV are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame are received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data is received and if address recognition has not rejected the frame, the receive FIFO signals the frame is accepted and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Therefore, no collision fragments are presented to you except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV, and TR status bits, and the frame length. See "Reception errors" for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is a babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See Ethernet Receive Buffer Descriptor (RxBD), for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame is received and is in memory. The Ethernet controller then waits for a new frame.

53.5.11 Ethernet address recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames appears in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 53-5 illustrates the address recognition decisions made by the receive block, while Figure 53-6 illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in Figure 53-5. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 53-6.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

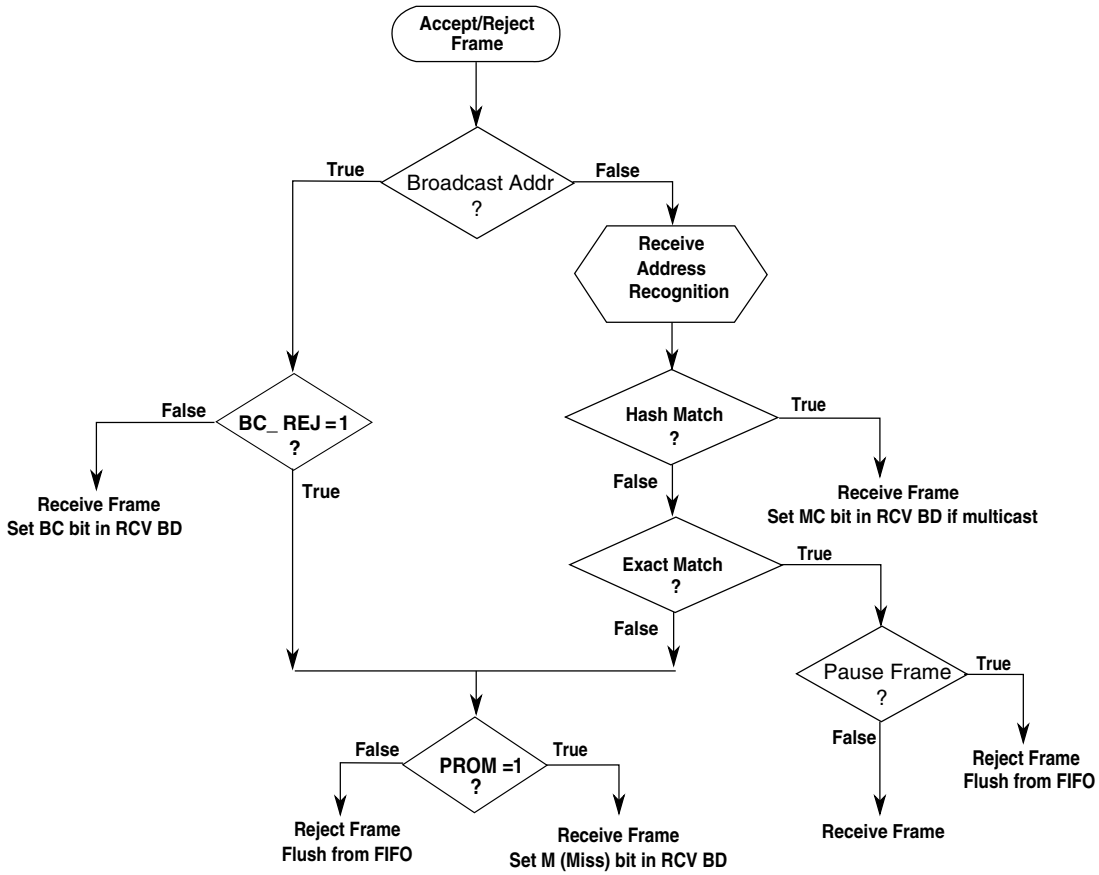
If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines the received frame is a valid PAUSE frame, the frame is rejected. The receiver detects a PAUSE frame with the DA field set to the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on PAUSE frame detection, shown in Figure 53-5.

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] set), the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

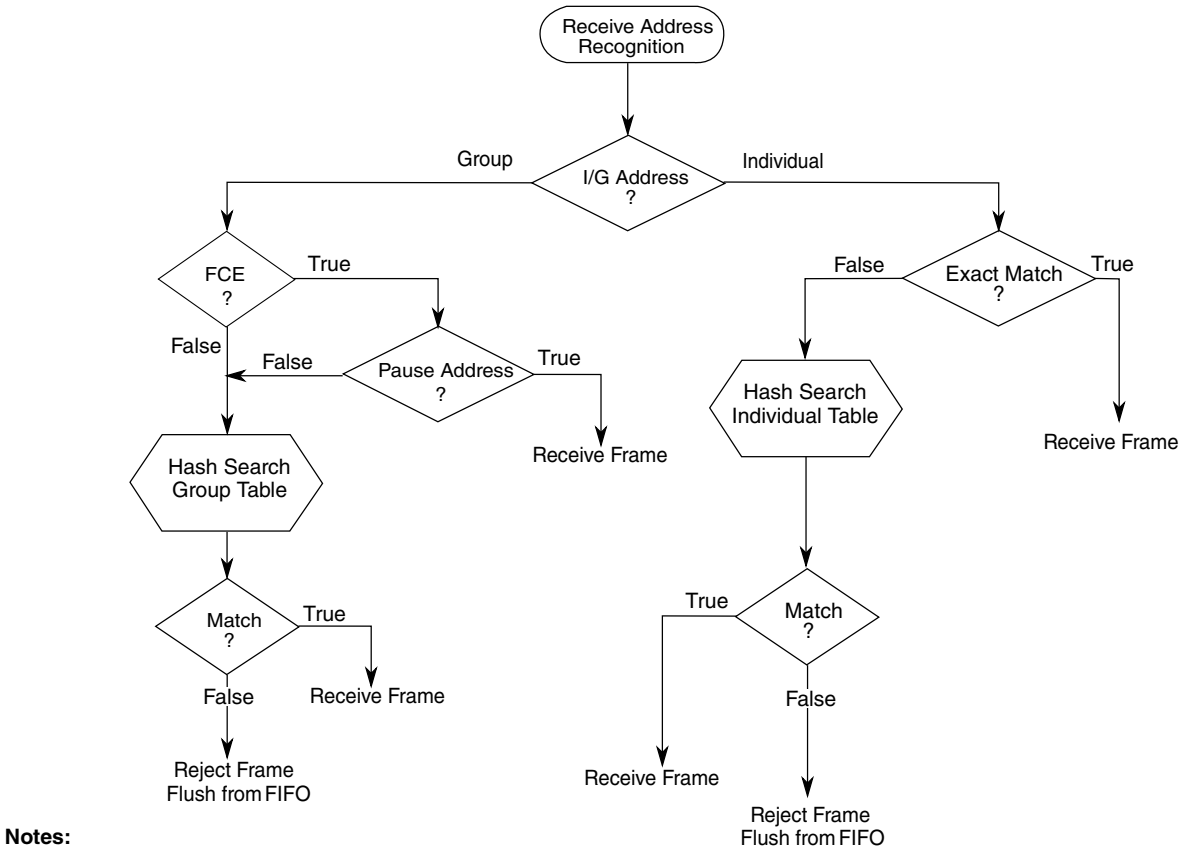
Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



Notes:
BC_REJ - field in RCR register (BroadCast REJect)
PROM - field in RCR register (PROMiscous mode)
Pause Frame - valid PAUSE frame received

Figure 53-5. Ethernet address recognition—receive block decisions



Notes:
FCE - field in RCR register (flow control enable)
I/G - Individual/Group bit in destination address (lsb in first byte received in MAC frame)

Figure 53-6. Ethernet address recognition—microcode decisions

53.5.12 Hash algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb = 1) or GALR (msb = 0). The five least significant bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

$$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$$

Equation 31. CRC32 polynomial for hash computation

The following table contains example destination addresses and corresponding hash values.

Table 53-16. Destination address to 6-bit hash

48-BIT DA	6-bit hash (in hex)	Hash decimal value
65FF_FFFF_FFFF	0h	0
55FF_FFFF_FFFF	1h	1
15FF_FFFF_FFFF	2h	2
35FF_FFFF_FFFF	3h	3
B5FF_FFFF_FFFF	4h	4
95FF_FFFF_FFFF	5h	5
D5FF_FFFF_FFFF	6h	6
F5FF_FFFF_FFFF	7h	7
DBFF_FFFF_FFFF	8h	8
FBFF_FFFF_FFFF	9h	9
BBFF_FFFF_FFFF	Ah	10
8BFF_FFFF_FFFF	Bh	11
0BFF_FFFF_FFFF	Ch	12
3BFF_FFFF_FFFF	Dh	13
7BFF_FFFF_FFFF	Eh	14
5BFF_FFFF_FFFF	Fh	15
27FF_FFFF_FFFF	10h	16
07FF_FFFF_FFFF	11h	17
57FF_FFFF_FFFF	12h	18
77FF_FFFF_FFFF	13h	19
F7FF_FFFF_FFFF	14h	20
C7FF_FFFF_FFFF	15h	21
97FF_FFFF_FFFF	16h	22
A7FF_FFFF_FFFF	17h	23
99FF_FFFF_FFFF	18h	24
B9FF_FFFF_FFFF	19h	25
F9FF_FFFF_FFFF	1Ah	26
C9FF_FFFF_FFFF	1Bh	27

Table continues on the next page...

Table 53-16. Destination address to 6-bit hash (continued)

48-BIT DA	6-bit hash (in hex)	Hash decimal value
59FF_FFFF_FFFF	1Ch	28
79FF_FFFF_FFFF	1Dh	29
29FF_FFFF_FFFF	1Eh	30
19FF_FFFF_FFFF	1Fh	31
D1FF_FFFF_FFFF	20h	32
F1FF_FFFF_FFFF	21h	33
B1FF_FFFF_FFFF	22h	34
91FF_FFFF_FFFF	23h	35
11FF_FFFF_FFFF	24h	36
31FF_FFFF_FFFF	25h	37
71FF_FFFF_FFFF	26h	38
51FF_FFFF_FFFF	27h	39
7FFF_FFFF_FFFF	28h	40
4FFF_FFFF_FFFF	29h	41
1FFF_FFFF_FFFF	2Ah	42
3FFF_FFFF_FFFF	2Bh	43
BFFF_FFFF_FFFF	2Ch	44
9FFF_FFFF_FFFF	2Dh	45
DFFF_FFFF_FFFF	2Eh	46
EFFF_FFFF_FFFF	2Fh	47
93FF_FFFF_FFFF	30h	48
B3FF_FFFF_FFFF	31h	49
F3FF_FFFF_FFFF	32h	50
D3FF_FFFF_FFFF	33h	51
53FF_FFFF_FFFF	34h	52
73FF_FFFF_FFFF	35h	53
23FF_FFFF_FFFF	36h	54
13FF_FFFF_FFFF	37h	55
3DFF_FFFF_FFFF	38h	56
0DFF_FFFF_FFFF	39h	57
5DFF_FFFF_FFFF	3Ah	58
7DFF_FFFF_FFFF	3Bh	59
FDFF_FFFF_FFFF	3Ch	60
DDFF_FFFF_FFFF	3Dh	61
9DFF_FFFF_FFFF	3Eh	62
BDFF_FFFF_FFFF	3Fh	63

53.5.13 Full duplex flow control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable PAUSE frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] set) with flow control (RCR[FCE] set). The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in the following table. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 53-17. PAUSE frame field specification

48-bit destination address	0180_C200_0001h or Physical Address
48-bit source address	Any
16-bit type	8808h
16-bit opcode	0001h
16-bit PAUSE duration	0000h — FFFFh

The receiver and microcontroller modules perform PAUSE frame detection. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is set by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. The pause timer uses the transmit backoff timer hardware for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing MAC data frame transmission to resume. The receive flow control pause status bit (TCR[RFC_PAUSE]) is set while the transmitter pauses due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and you must set flow control pause (TCR[TFC_PAUSE]). After TCR[TFC_PAUSE] is set, the transmitter sets TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts and the pause frame is transmitted. TCR[TFC_PAUSE,GTS] are then cleared internally.

You must specify the desired pause duration in the OPD register.

When the transmitter pauses due to receiver/microcontroller pause frame detection, TCR[TFC_PAUSE] may remain set and cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

53.5.14 Inter-Packet Gap (IPG) time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver accepts back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the receiver may discard the following frame.

53.5.15 Collision managing

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the sequence, a JAM pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit times (one slot time), the retry process is initiated. The transmitter waits a random number of slot times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

53.5.16 MII internal and external loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the RCR[LOOP, DRT] and TCR[FDEN] bits.

Set FDEN for internal and external loopback.

For internal loopback, set RCR[LOOP] and clear RCR[DRT]. FEC_TXEN and FEC_TXER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the transmit and receive blocks use the internal bus clock instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit

and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underruns and receive FIFO overflows.

For external loopback, clear RCR[LOOP] and RCR[DRT], and configure the external transceiver for loopback.

53.5.17 RMII loopback

The FEC also supports RMII loopback. In this mode, transmit data is sent to the RMII receive logic and out the FEC_TXD[1:0] pins. To enable RMII loopback mode, set RCR[RMII_MODE, RMII_LOOP] and TCR[FDEN] and clear RCR[RMII_ECHO, LOOP].

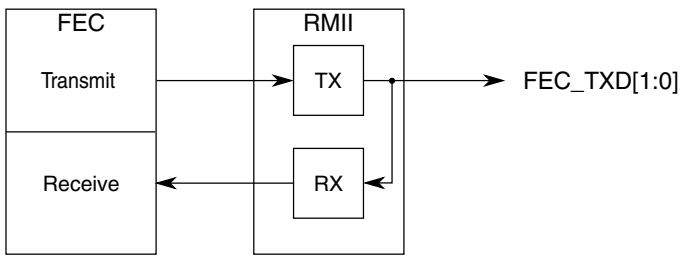


Figure 53-7. RMII loopback mode

53.5.18 RMII echo

The Ethernet controller also supports RMII echo mode, which transmits data on FEC_TXD[1:0] as it is received on FEC_RXD[1:0]. To enable RMII echo mode, set RCR[RMII_MODE, RMII_ECHO] bits and clear RCR[RMII_LOOP, LOOP].

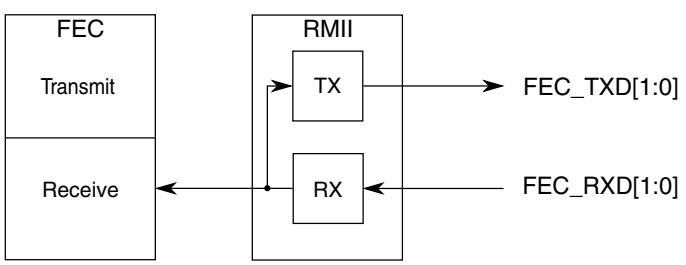


Figure 53-8. RMII echo mode

53.5.19 Ethernet error-managing procedure

The Ethernet controller reports frame reception and transmission error conditions using the MIB block counters, the FEC RxBDs, and the EIR register.

53.5.19.1 Transmission errors

53.5.19.1.1 Transmitter underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed, and EIR[UN] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The UN interrupt is asserted if enabled in the EIMR register.

53.5.19.1.2 Retransmission attempts limit expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[RL] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The RL interrupt is asserted if enabled in the EIMR register.

53.5.19.1.3 Late collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[LC] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The LC interrupt is asserted if enabled in the EIMR register.

53.5.19.1.4 Heartbeat

Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within four microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver continues to function properly. This is the heartbeat condition.

If TCR[HBC] is set and the heartbeat condition is not detected by the FEC after a frame transmission, a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets EIR[HB], and generates the HBERR interrupt if it is enabled.

53.5.19.2 Reception errors

53.5.19.2.1 Overrun error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, FEC sets RxBD[OV]. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. The driver must discard this frame.

53.5.19.2.2 Non-octet error (dribbling bits)

The Ethernet controller manages up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, no error is reported.

53.5.19.2.3 CRC error

When a CRC error occurs with no dribble bits, FEC closes the buffer and sets RxBD[CR]. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

53.5.19.2.4 Frame length violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and RxBD[LG] is set. The frame is not truncated unless the frame length exceeds 2047 bytes.

53.5.19.2.5 Truncation

When the receive frame length exceeds 2047 bytes, frame is truncated and RxBD[TR] is set.

Chapter 54
FlexRay Communication Controller (FlexRay)

54.1 Introduction

This section provides a high-level summary of module features.

54.1.1 Reference

The following documents are referenced.

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*[1]
  - Refer to this document for the all of the FlexRay related informaton including the configuration parameters and the allowed parameter ranges.
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0*

54.1.2 Glossary

This section provides a list of terms used in this chapter.

Table 54-1. List of terms

Term	Definition
BCU	Buffer Control Unit. Handles message buffer access.
BMIF	Bus Master Interface. Provides master access to FlexRay memory area.
CC	Communication Controller
CDC	Clock Domain Crosser
CHI	Controller Host Interface

Table continues on the next page...

1. The FlexRay Specifications have been developed for automotive applications.The FlexRay Specifications have been neither developed nor tested for non-automotive applications.

Table 54-1. List of terms (continued)

Term	Definition
Cycle length in µT	The actual length of a cycle in µT for the ideal controller (+/- 0 ppm)
EBI	External Bus Interface
FlexRay Memory Area	Memory area to store the physical message buffer payload data, frame header, frame and slot status, and synchronization frame related tables.
FSS	Frame Start Sequence
HIF	Host Interface. Provides host access to controller.
Host	The FlexRay CC host MCU
Keyslot	Key slot is used to transmit the startup frame, sync frame, or designated single slot frame.
LUT	Look Up Table. Stores message buffer header index value.
LRAM	Look Up Table RAM. Module internal memory to store message buffer configuration data and data field offsets for individual message buffers and receive shadow buffers.
MB	Message Buffer
Mini-slot	An interval of time within the dynamic segment of the communication cycle that is of constant duration (in terms of macroticks) and that is used by the synchronized FTDMA media access scheme to manage media arbitration
MBIDX	Message Buffer Index: the position of a header field entry within the header area. If the header area is accessed as an array, this is the same as the array index of the entry.
MBNum	Message Buffer Number: Position of message buffer configuration registers within the register map. For example, Message Buffer Number 5 corresponds to the MBCCS5 register.
MCU	Microcontroller Unit
µT	Microtick
MT	Macrotick
MTS	Media Access Test Symbol
message frame	Frame with *Null Frame Indicator* set to 1
normal frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 0
null frame	frame with *Null Frame Indicator* set to 0
NIT	Network Idle Time
PE	Protocol Engine
POC	Protocol Operation Control. Each state of the POC is denoted by POC:state
Rx	Reception
Slot	An interval of time during which access to a communication channel is granted exclusively to a specific node for the transmission of a frame with a frame ID corresponding to the slot.
SEQ	Sequencer Engine
SU	Status update
SECDED	Single-bit error correction, double-bit error detection
System Memory	Memory that contains the FlexRay Memory Area.
System Bus	Bus that connects the controller and System Memory

Table continues on the next page...

Table 54-1. List of terms (continued)

Term	Definition
sync frame	null frame or message frame with *Sync Frame Indicator* set to 1
startup frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 1
TCU	Time Control Unit
Tx	Transmission

54.1.3 Overview

The CC is a FlexRay communication controller that implements the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

The CC has three main components:

- *Controller host interface (CHI)*
- *Protocol engine (PE)*
- *Clock domain crossing unit (CDC)*

A block diagram of the CC with its surrounding modules is given in the below figure.

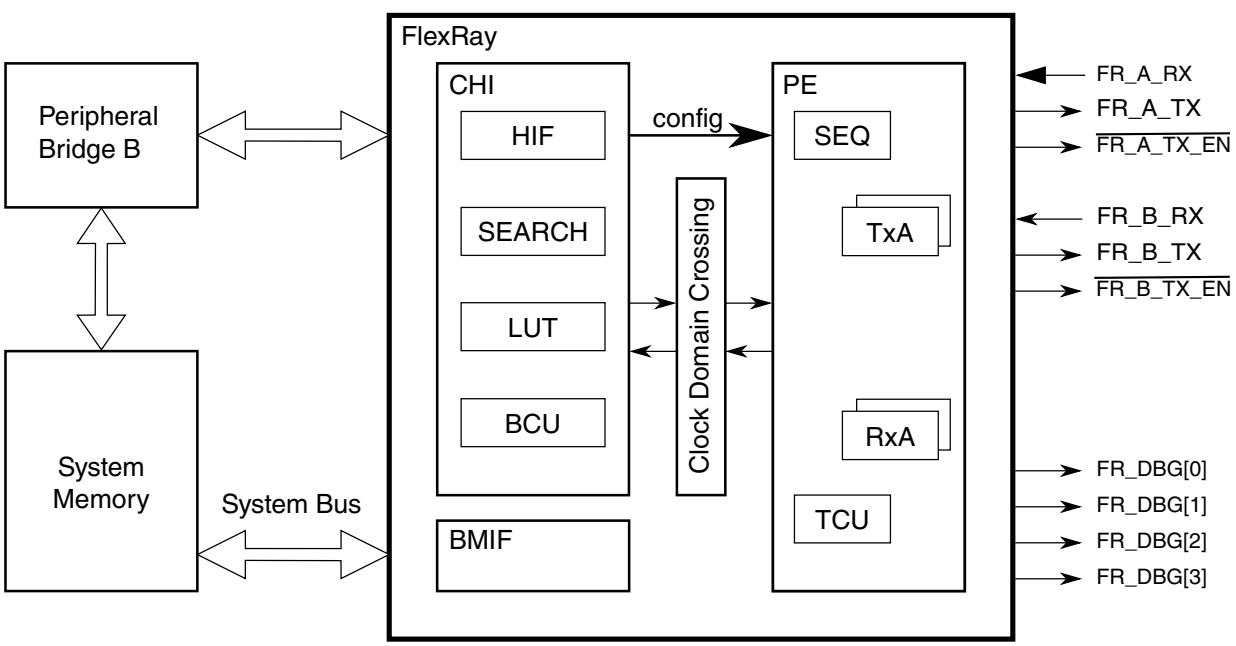


Figure 54-1. FlexRay block diagram

The protocol engine has two transmitter units TxA and TxB and two receiver units RxA and RxB for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the PE is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, and the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the CHI clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the System Memory Base Address Register (FR_SYMBADHR and FR_SYMBADLR). All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note

The CC does not provide a memory protection scheme for the FlexRay memory area.

54.1.4 Features

The CC provides the following features:

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A compliant protocol implementation*
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0 compliant bus driver interface*

- Single channel support
  - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B.
- Dual channel support
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 128 configurable message buffers with
  - individual frame ID filtering
  - individual channel ID filtering
  - individual cycle counter filtering
- Message buffer header, status and payload data stored in dedicated FlexRay memory area
  - allows for flexible and efficient message buffer implementation
  - consistent data access ensured by means of buffer locking scheme
  - application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0 up to 254 bytes
- Two independent message buffer segments with configurable size of payload data section
  - each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
  - applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
  - receive message buffer
  - transmit message buffer
- Individual message buffer reconfiguration supported
  - means provided to safely disable individual message buffers
  - disabled message buffers can be reconfigured

- Two independent receive FIFOs
  - one receive FIFO per channel
  - up to 255 entries for each FIFO
  - global frame ID filtering, based on both value/mask filters and range filters
  - global channel ID filtering
  - global message ID filtering for the dynamic segment
- 4 configurable slot error counters
- 4 dedicated slot status indicators
  - used to observe slots without using receive message buffers
- Measured value indicators for the clock synchronization
  - internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- 1 absolute timer
- 1 timer that can be configured to absolute or relative
- Error correction and error detection (SECDED ECC) for protocol engine data RAM
- Error detection (SECDED ECC) for CHI lookup table RAM

54.1.5 Modes of Operation

This section describes the basic operational power modes of the CC.

54.1.5.1 Disabled Mode

The CC enters the Disabled Mode during hard reset or when the host clears the ‘MEN’ field in the Module Configuration Register (FR_MCR). The host can clear this field by writing ‘0’ and only when the module is in POC:default config mode. The Disabled mode can be checked by reading the module enable field, MEN, in the Module Configuration Register (FR_MCR).

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled Mode* can be accessed for writing as stated in the Memory Map and Register Description section.

The application configures the CC by accessing the configuration bits and fields in the Module Configuration Register (FR_MCR) as described in Module Initialization .

54.1.5.1.1 Leave Disabled Mode

The CC leaves the Disabled Mode and enters the Normal Mode, when the application writes 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

Note

Once the CC is enabled, it can only be disabled during POC: default config.

54.1.5.2 Normal Mode

In this mode the CC is fully functional. The CC indicates that it is in Normal Mode by asserting the module enable bit MEN in the Module Configuration Register (FR_MCR).

54.1.5.2.1 Enter Normal Mode

This mode is entered when the application requests the CC to leave the Disabled Mode. If the Normal Mode was entered by leaving the Disabled Mode, the application has to perform the protocol initialization described in Protocol Initialization to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB bits in the Module Configuration Register (FR_MCR), the corresponding FlexRay bus driver ports are enabled and driven.

54.2 External Signal Description

This section lists and describes the CC signals connected to external pins. These signals are summarized in the following table and described in detail in the Detailed Signal Descriptions section below.

NOTE

Please refer the device configuration of the reference manual for the exact module pin names.

NOTE

The off chip signals FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 54-2. External Signal Properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
$\overline{\text{FR\_A\_TX\_EN}}$	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
$\overline{\text{FR\_B\_TX\_EN}}$	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

54.2.1 Detailed Signal Descriptions

This section provides a detailed description of the CC signals, connected to external pins.

54.2.1.1 FR_A_RX — Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

54.2.1.2 FR_A_TX — Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

54.2.1.3 $\overline{\text{FR\_A\_TX\_EN}}$ — Transmit Enable Channel A

The $\overline{\text{FR\_A\_TX\_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

54.2.1.4 FR_B_RX — Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

54.2.1.5 FR_B_TX — Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver

54.2.1.6 $\overline{\text{FR\_B\_TX\_EN}}$ — Transmit Enable Channel B

The $\overline{\text{FR\_B\_TX\_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

54.2.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0] — Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection refer to the Strobe Signal Support.

54.3 Controller Host Interface Clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

The first constraint corresponds to the number of utilized message buffers and is specified in Number of Usable Message Buffers.

The second constraint corresponds to the value of the TIMEOUT field in the System Memory Access Time-Out Register (FR_SYMATOR) and is specified in Configure System Memory Access Time-Out Register (FR_SYMATOR).

54.4 Protocol Engine Clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select bit CLKSEL in the Module Configuration Register (FR_MCR).

Note

See the Device clocking details for the exact clock sources used.

54.4.1 Oscillator Clocking

If the protocol engine is clocked by the internal crystal oscillator, a crystal or CMOS compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version* 2.1 Rev A.

54.4.2 PLL Clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately and provided to the PE.

For more details, see the clocking chapter of the device reference manual.

54.5 Register Descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities.

Table 54-3. Register Access Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
R*	Reserved bit or field, will not be changed. Application must not write any value different from the reset value.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 54-4. Register Field Types

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect.
Reset Value	
0	Resets to zero.
1	Resets to one.
-	Not defined after reset and not affected by reset.

54.5.1 Register Reset

All registers except the Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*), Message Buffer Frame ID Register (FR_MBFIDR*n*), and Message Buffer Index Register (FR_MBIDXR*n*) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are mentioned in the detailed description of the register. The additional reset conditions are explained in the table below.

Table 54-5. Additional Register Reset Conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes to RUN command "0101" to the POCCMD field in the Protocol Operation Control Register (FR_POCR).
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants the disable to the application by clearing the FR_MBCCSRn[EDS] bit.

54.5.2 Register Write Access

This section describes the write access restriction terms that apply to all registers.

54.5.2.1 Register Write Access Restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the table below. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 54-6. Register Write Access Restrictions

Condition	Indication	Description
Any Time	-	No write access restriction.
Disabled Mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled Mode.
Normal Mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal Mode.
POC:config	FR_PSR0[PROTSTATE] = *POC:config*	Write access only when Protocol is in the *POC:config* state.
MB_DIS	FR_MBCCSR[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.

54.5.2.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is described in the Memory map and register definition section. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

54.5.2.3 Internal Register Access

The following memory mapped registers are used to access multiple internal registers.

- *Strobe Signal Control Register (FR_STBSCR)*
- *Slot Status Selection Register (FR_SSSR)*

- *Slot Status Counter Condition Register (FR_SSCCR)*
- *Receive Shadow Buffer Configuration Data*

Each of these memory mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If the WMD bit is set to 0 during the write access, all fields of the internal register are updated. If the WMD bit set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

54.6 Memory map and register definition

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU.

NOTE

The following registers are 16-bit write accessible only.

Strobe Signal Control Register (FR_STBSCR)

PE DRAM Access Register (FR_PEDRAR)

PE DRAM Data Register (FR_PEDRDR)

Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Slot Status Selection Register (FR_SSSR)

Slot Status Counter Condition Register (FR_SSCCR)

Receive Shadow Buffer Index Register (FR_RSBIR)

Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Message Buffer Cycle Counter Filter Register (FR_MBCCFRn)

Message Buffer Frame ID Register (FR_MBFIDRn)

Message Buffer Index Register (FR_MBIDXRn)

Message Buffer Data Field Offset Register (FR_MBDORn)

LRAM ECC Error Test Register (FR_LEETRn)

FR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Version Register (FR_MVR)	16	R	A568h	54.6.1/2477
2	Module Configuration Register (FR_MCR)	16	R/W	0000h	54.6.2/2477
4	System Memory Base Address High Register (FR_SYMBADHR)	16	R/W	0000h	54.6.3/2480
6	System Memory Base Address Low Register (FR_SYMBADLR)	16	R/W	0000h	54.6.4/2481
8	Strobe Signal Control Register (FR_STBSCR)	16	R/W	0000h	54.6.5/2481
C	Message Buffer Data Size Register (FR_MBDSR)	16	R/W	0000h	54.6.6/2483
E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	16	R/W	7F7Fh	54.6.7/2484
10	PE DRAM Access Register (FR_PEDRAR)	16	R/W	0000h	54.6.8/2485
12	PE DRAM Data Register (FR_PEDRDR)	16	R/W	0000h	54.6.9/2486
14	Protocol Operation Control Register (FR_POCR)	16	R/W	0000h	54.6.10/ 2486
16	Global Interrupt Flag and Enable Register (FR_GIFER)	16	R/W	0000h	54.6.11/ 2488
18	Protocol Interrupt Flag Register 0 (FR_PIFR0)	16	R/W	0000h	54.6.12/ 2491
1A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	16	R/W	0000h	54.6.13/ 2493
1C	Protocol Interrupt Enable Register 0 (FR_PIER0)	16	R/W	0000h	54.6.14/ 2495
1E	Protocol Interrupt Enable Register 1 (FR_PIER1)	16	R/W	0000h	54.6.15/ 2497
20	CHI Error Flag Register (FR_CHIERFR)	16	R/W	0000h	54.6.16/ 2498
22	Message Buffer Interrupt Vector Register (FR_MBIVEC)	16	R	0000h	54.6.17/ 2501
24	Channel A Status Error Counter Register (FR_CASERCR)	16	R	0000h	54.6.18/ 2502
26	Channel B Status Error Counter Register (FR_CBSERCR)	16	R	0000h	54.6.19/ 2502
28	Protocol Status Register 0 (FR_PSR0)	16	R	0000h	54.6.20/ 2503
2A	Protocol Status Register 1 (FR_PSR1)	16	R	0000h	54.6.21/ 2505
2C	Protocol Status Register 2 (FR_PSR2)	16	R	0000h	54.6.22/ 2506
2E	Protocol Status Register 3 (FR_PSR3)	16	R/W	0000h	54.6.23/ 2508
30	Macrotick Counter Register (FR_MTCTR)	16	R	0000h	54.6.24/ 2510
32	Cycle Counter Register (FR_CYCTR)	16	R	0000h	54.6.25/ 2511

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Slot Counter Channel A Register (FR_SLTCTAR)	16	R	0000h	54.6.26/ 2511
36	Slot Counter Channel B Register (FR_SLTCTBR)	16	R	0000h	54.6.27/ 2512
38	Rate Correction Value Register (FR_RTCORVR)	16	R	0000h	54.6.28/ 2512
3A	Offset Correction Value Register (FR_OFCORVR)	16	R	0000h	54.6.29/ 2513
3C	Combined Interrupt Flag Register (FR_CIFR)	16	R	0000h	54.6.30/ 2514
3E	System Memory Access Time-Out Register (FR_SYMATOR)	16	R/W	0006h	54.6.31/ 2515
40	Sync Frame Counter Register (FR_SFCNTR)	16	R	0000h	54.6.32/ 2516
42	Sync Frame Table Offset Register (FR_SFTOR)	16	R/W	0000h	54.6.33/ 2516
44	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	16	R/W	0000h	54.6.34/ 2517
46	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	16	R/W	0000h	54.6.35/ 2519
48	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	16	R/W	0000h	54.6.36/ 2519
4A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	16	R/W	0000h	54.6.37/ 2520
4C	Network Management Vector Register (FR_NMVR0)	16	R	0000h	54.6.38/ 2520
4E	Network Management Vector Register (FR_NMVR1)	16	R	0000h	54.6.38/ 2520
50	Network Management Vector Register (FR_NMVR2)	16	R	0000h	54.6.38/ 2520
52	Network Management Vector Register (FR_NMVR3)	16	R	0000h	54.6.38/ 2520
54	Network Management Vector Register (FR_NMVR4)	16	R	0000h	54.6.38/ 2520
56	Network Management Vector Register (FR_NMVR5)	16	R	0000h	54.6.38/ 2520
58	Network Management Vector Length Register (FR_NMVLR)	16	R/W	0000h	54.6.39/ 2521
5A	Timer Configuration and Control Register (FR_TICCR)	16	R/W	0000h	54.6.40/ 2521
5C	Timer 1 Cycle Set Register (FR_TI1CYSR)	16	R/W	0000h	54.6.41/ 2523
5E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	16	R/W	0000h	54.6.42/ 2524

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)	16	R/W	0000h	54.6.43/ 2524
60	Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)	16	R/W	0000h	54.6.44/ 2525
62	Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)	16	R/W	0000h	54.6.45/ 2525
62	Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)	16	R/W	0000h	54.6.46/ 2526
64	Slot Status Selection Register (FR_SSSR)	16	R/W	0000h	54.6.47/ 2527
66	Slot Status Counter Condition Register (FR_SSCCR)	16	R/W	0000h	54.6.48/ 2528
68	Slot Status Register (FR_SSR0)	16	R	0000h	54.6.49/ 2530
6A	Slot Status Register (FR_SSR1)	16	R	0000h	54.6.49/ 2530
6C	Slot Status Register (FR_SSR2)	16	R	0000h	54.6.49/ 2530
6E	Slot Status Register (FR_SSR3)	16	R	0000h	54.6.49/ 2530
70	Slot Status Register (FR_SSR4)	16	R	0000h	54.6.49/ 2530
72	Slot Status Register (FR_SSR5)	16	R	0000h	54.6.49/ 2530
74	Slot Status Register (FR_SSR6)	16	R	0000h	54.6.49/ 2530
76	Slot Status Register (FR_SSR7)	16	R	0000h	54.6.49/ 2530
78	Slot Status Counter Register (FR_SSCR0)	16	R	0000h	54.6.50/ 2532
7A	Slot Status Counter Register (FR_SSCR1)	16	R	0000h	54.6.50/ 2532
7C	Slot Status Counter Register (FR_SSCR2)	16	R	0000h	54.6.50/ 2532
7E	Slot Status Counter Register (FR_SSCR3)	16	R	0000h	54.6.50/ 2532
80	MTS A Configuration Register (FR_MTSACFR)	16	R/W	0000h	54.6.51/ 2532
82	MTS B Configuration Register (FR_MTSBCFR)	16	R/W	0000h	54.6.52/ 2533
84	Receive Shadow Buffer Index Register (FR_RSBIR)	16	R/W	0000h	54.6.53/ 2534
86	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	16	R/W	0000h	54.6.54/ 2535

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	Receive FIFO Start Index Register (FR_RFSIR)	16	R/W	0000h	54.6.55/ 2536
8A	Receive FIFO Depth and Size Register (FR_RFDSR)	16	R/W	0000h	54.6.56/ 2536
8C	Receive FIFO A Read Index Register (FR_RFARIR)	16	R	0000h	54.6.57/ 2537
8E	Receive FIFO B Read Index Register (FR_RFBRIR)	16	R	0000h	54.6.58/ 2537
90	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	16	R/W	0000h	54.6.59/ 2538
92	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	16	R/W	0000h	54.6.60/ 2538
94	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	16	R/W	0000h	54.6.61/ 2539
96	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	16	R/W	0000h	54.6.62/ 2539
98	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	16	R/W	0000h	54.6.63/ 2540
9A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	16	R/W	0000h	54.6.64/ 2541
9C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	16	R	0000h	54.6.65/ 2542
9E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	16	R	0000h	54.6.66/ 2543
A0	Protocol Configuration Register 0 (FR_PCR0)	16	R/W	0000h	54.6.67/ 2543
A2	Protocol Configuration Register 1 (FR_PCR1)	16	R/W	0000h	54.6.68/ 2546
A4	Protocol Configuration Register 2 (FR_PCR2)	16	R/W	0000h	54.6.69/ 2546
A6	Protocol Configuration Register 3 (FR_PCR3)	16	R/W	0000h	54.6.70/ 2547
A8	Protocol Configuration Register 4 (FR_PCR4)	16	R/W	0000h	54.6.71/ 2547
AA	Protocol Configuration Register 5 (FR_PCR5)	16	R/W	0000h	54.6.72/ 2548
AC	Protocol Configuration Register 6 (FR_PCR6)	16	R/W	0000h	54.6.73/ 2548
AE	Protocol Configuration Register 7 (FR_PCR7)	16	R/W	0000h	54.6.74/ 2549
B0	Protocol Configuration Register 8 (FR_PCR8)	16	R/W	0000h	54.6.75/ 2549
B2	Protocol Configuration Register 9 (FR_PCR9)	16	R/W	0000h	54.6.76/ 2550

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B4	Protocol Configuration Register 10 (FR_PCR10)	16	R/W	0000h	54.6.77/ 2551
B6	Protocol Configuration Register 11 (FR_PCR11)	16	R/W	0000h	54.6.78/ 2551
B8	Protocol Configuration Register 12 (FR_PCR12)	16	R/W	0000h	54.6.79/ 2552
BA	Protocol Configuration Register 13 (FR_PCR13)	16	R/W	0000h	54.6.80/ 2553
BC	Protocol Configuration Register 14 (FR_PCR14)	16	R/W	0000h	54.6.81/ 2553
BE	Protocol Configuration Register 15 (FR_PCR15)	16	R/W	0000h	54.6.82/ 2554
C0	Protocol Configuration Register 16 (FR_PCR16)	16	R/W	0000h	54.6.83/ 2554
C2	Protocol Configuration Register 17 (FR_PCR17)	16	R/W	0000h	54.6.84/ 2555
C4	Protocol Configuration Register 18 (FR_PCR18)	16	R/W	0000h	54.6.85/ 2555
C6	Protocol Configuration Register 19 (FR_PCR19)	16	R/W	0000h	54.6.86/ 2556
C8	Protocol Configuration Register 20 (FR_PCR20)	16	R/W	0000h	54.6.87/ 2556
CA	Protocol Configuration Register 21 (FR_PCR21)	16	R/W	0000h	54.6.88/ 2557
CC	Protocol Configuration Register 22 (FR_PCR22)	16	R/W	0000h	54.6.89/ 2557
CE	Protocol Configuration Register 23 (FR_PCR23)	16	R/W	0000h	54.6.90/ 2558
D0	Protocol Configuration Register 24 (FR_PCR24)	16	R/W	0000h	54.6.91/ 2558
D2	Protocol Configuration Register 25 (FR_PCR25)	16	R/W	0000h	54.6.92/ 2559
D4	Protocol Configuration Register 26 (FR_PCR26)	16	R/W	0000h	54.6.93/ 2559
D6	Protocol Configuration Register 27 (FR_PCR27)	16	R/W	0000h	54.6.94/ 2560
D8	Protocol Configuration Register 28 (FR_PCR28)	16	R/W	0000h	54.6.95/ 2561
DA	Protocol Configuration Register 29 (FR_PCR29)	16	R/W	0000h	54.6.96/ 2561
DC	Protocol Configuration Register 30 (FR_PCR30)	16	R/W	0000h	54.6.97/ 2562
DE	StopWatch Count Register (FR_STPWR)	32	R/W	0000_0000h	54.6.98/ 2562

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E2	Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)	16	R/W	0000h	54.6.99/ 2563
E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	16	R/W	0000h	54.6.100/ 2563
E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	16	R/W	0000h	54.6.101/ 2564
EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	16	R/W	0000h	54.6.102/ 2565
EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	16	R/W	0000h	54.6.103/ 2565
EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	16	R/W	0000h	54.6.104/ 2566
F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	16	R/W	0000h	54.6.105/ 2567
F2	ECC Error Report and Injection Control Register (FR_EERICR)	16	R/W	0000h	54.6.106/ 2569
F4	ECC Error Report Address Register (FR_EERAR)	16	R	7000h	54.6.107/ 2570
F6	ECC Error Report Data Register (FR_EERDR)	16	R	0000h	54.6.108/ 2571
F8	ECC Error Report Code Register (FR_EERCR)	16	R	0000h	54.6.109/ 2572
FA	ECC Error Injection Address Register (FR_EEIAR)	16	R/W	0000h	54.6.110/ 2573
FC	ECC Error Injection Data Register (FR_EEIDR)	16	R/W	0000h	54.6.111/ 2573
FE	ECC Error Injection Code Register (FR_EEICR)	16	R/W	0000h	54.6.112/ 2574
800	Message Buffer Configuration, Control, Status Register (FR_MBCCSR0)	16	R/W	0000h	54.6.113/ 2574
802	Message Buffer Cycle Counter Filter Register (FR_MBCCFR0)	16	R/W	See section	54.6.114/ 2576
804	Message Buffer Frame ID Register (FR_MBFIDR0)	16	R/W	See section	54.6.115/ 2578
806	Message Buffer Index Register (FR_MBIDXR0)	16	R/W	See section	54.6.116/ 2579
808	Message Buffer Configuration, Control, Status Register (FR_MBCCSR1)	16	R/W	0000h	54.6.113/ 2574
80A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR1)	16	R/W	See section	54.6.114/ 2576
80C	Message Buffer Frame ID Register (FR_MBFIDR1)	16	R/W	See section	54.6.115/ 2578
80E	Message Buffer Index Register (FR_MBIDXR1)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
810	Message Buffer Configuration, Control, Status Register (FR_MBCCSR2)	16	R/W	0000h	54.6.113/ 2574
812	Message Buffer Cycle Counter Filter Register (FR_MBCCFR2)	16	R/W	See section	54.6.114/ 2576
814	Message Buffer Frame ID Register (FR_MBFIDR2)	16	R/W	See section	54.6.115/ 2578
816	Message Buffer Index Register (FR_MBIDXR2)	16	R/W	See section	54.6.116/ 2579
818	Message Buffer Configuration, Control, Status Register (FR_MBCCSR3)	16	R/W	0000h	54.6.113/ 2574
81A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR3)	16	R/W	See section	54.6.114/ 2576
81C	Message Buffer Frame ID Register (FR_MBFIDR3)	16	R/W	See section	54.6.115/ 2578
81E	Message Buffer Index Register (FR_MBIDXR3)	16	R/W	See section	54.6.116/ 2579
820	Message Buffer Configuration, Control, Status Register (FR_MBCCSR4)	16	R/W	0000h	54.6.113/ 2574
822	Message Buffer Cycle Counter Filter Register (FR_MBCCFR4)	16	R/W	See section	54.6.114/ 2576
824	Message Buffer Frame ID Register (FR_MBFIDR4)	16	R/W	See section	54.6.115/ 2578
826	Message Buffer Index Register (FR_MBIDXR4)	16	R/W	See section	54.6.116/ 2579
828	Message Buffer Configuration, Control, Status Register (FR_MBCCSR5)	16	R/W	0000h	54.6.113/ 2574
82A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR5)	16	R/W	See section	54.6.114/ 2576
82C	Message Buffer Frame ID Register (FR_MBFIDR5)	16	R/W	See section	54.6.115/ 2578
82E	Message Buffer Index Register (FR_MBIDXR5)	16	R/W	See section	54.6.116/ 2579
830	Message Buffer Configuration, Control, Status Register (FR_MBCCSR6)	16	R/W	0000h	54.6.113/ 2574
832	Message Buffer Cycle Counter Filter Register (FR_MBCCFR6)	16	R/W	See section	54.6.114/ 2576
834	Message Buffer Frame ID Register (FR_MBFIDR6)	16	R/W	See section	54.6.115/ 2578
836	Message Buffer Index Register (FR_MBIDXR6)	16	R/W	See section	54.6.116/ 2579
838	Message Buffer Configuration, Control, Status Register (FR_MBCCSR7)	16	R/W	0000h	54.6.113/ 2574
83A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR7)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
83C	Message Buffer Frame ID Register (FR_MBFIDR7)	16	R/W	See section	54.6.115/ 2578
83E	Message Buffer Index Register (FR_MBIDXR7)	16	R/W	See section	54.6.116/ 2579
840	Message Buffer Configuration, Control, Status Register (FR_MBCCSR8)	16	R/W	0000h	54.6.113/ 2574
842	Message Buffer Cycle Counter Filter Register (FR_MBCCFR8)	16	R/W	See section	54.6.114/ 2576
844	Message Buffer Frame ID Register (FR_MBFIDR8)	16	R/W	See section	54.6.115/ 2578
846	Message Buffer Index Register (FR_MBIDXR8)	16	R/W	See section	54.6.116/ 2579
848	Message Buffer Configuration, Control, Status Register (FR_MBCCSR9)	16	R/W	0000h	54.6.113/ 2574
84A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR9)	16	R/W	See section	54.6.114/ 2576
84C	Message Buffer Frame ID Register (FR_MBFIDR9)	16	R/W	See section	54.6.115/ 2578
84E	Message Buffer Index Register (FR_MBIDXR9)	16	R/W	See section	54.6.116/ 2579
850	Message Buffer Configuration, Control, Status Register (FR_MBCCSR10)	16	R/W	0000h	54.6.113/ 2574
852	Message Buffer Cycle Counter Filter Register (FR_MBCCFR10)	16	R/W	See section	54.6.114/ 2576
854	Message Buffer Frame ID Register (FR_MBFIDR10)	16	R/W	See section	54.6.115/ 2578
856	Message Buffer Index Register (FR_MBIDXR10)	16	R/W	See section	54.6.116/ 2579
858	Message Buffer Configuration, Control, Status Register (FR_MBCCSR11)	16	R/W	0000h	54.6.113/ 2574
85A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR11)	16	R/W	See section	54.6.114/ 2576
85C	Message Buffer Frame ID Register (FR_MBFIDR11)	16	R/W	See section	54.6.115/ 2578
85E	Message Buffer Index Register (FR_MBIDXR11)	16	R/W	See section	54.6.116/ 2579
860	Message Buffer Configuration, Control, Status Register (FR_MBCCSR12)	16	R/W	0000h	54.6.113/ 2574
862	Message Buffer Cycle Counter Filter Register (FR_MBCCFR12)	16	R/W	See section	54.6.114/ 2576
864	Message Buffer Frame ID Register (FR_MBFIDR12)	16	R/W	See section	54.6.115/ 2578
866	Message Buffer Index Register (FR_MBIDXR12)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
868	Message Buffer Configuration, Control, Status Register (FR_MBCCSR13)	16	R/W	0000h	54.6.113/ 2574
86A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR13)	16	R/W	See section	54.6.114/ 2576
86C	Message Buffer Frame ID Register (FR_MBFIDR13)	16	R/W	See section	54.6.115/ 2578
86E	Message Buffer Index Register (FR_MBIDXR13)	16	R/W	See section	54.6.116/ 2579
870	Message Buffer Configuration, Control, Status Register (FR_MBCCSR14)	16	R/W	0000h	54.6.113/ 2574
872	Message Buffer Cycle Counter Filter Register (FR_MBCCFR14)	16	R/W	See section	54.6.114/ 2576
874	Message Buffer Frame ID Register (FR_MBFIDR14)	16	R/W	See section	54.6.115/ 2578
876	Message Buffer Index Register (FR_MBIDXR14)	16	R/W	See section	54.6.116/ 2579
878	Message Buffer Configuration, Control, Status Register (FR_MBCCSR15)	16	R/W	0000h	54.6.113/ 2574
87A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR15)	16	R/W	See section	54.6.114/ 2576
87C	Message Buffer Frame ID Register (FR_MBFIDR15)	16	R/W	See section	54.6.115/ 2578
87E	Message Buffer Index Register (FR_MBIDXR15)	16	R/W	See section	54.6.116/ 2579
880	Message Buffer Configuration, Control, Status Register (FR_MBCCSR16)	16	R/W	0000h	54.6.113/ 2574
882	Message Buffer Cycle Counter Filter Register (FR_MBCCFR16)	16	R/W	See section	54.6.114/ 2576
884	Message Buffer Frame ID Register (FR_MBFIDR16)	16	R/W	See section	54.6.115/ 2578
886	Message Buffer Index Register (FR_MBIDXR16)	16	R/W	See section	54.6.116/ 2579
888	Message Buffer Configuration, Control, Status Register (FR_MBCCSR17)	16	R/W	0000h	54.6.113/ 2574
88A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR17)	16	R/W	See section	54.6.114/ 2576
88C	Message Buffer Frame ID Register (FR_MBFIDR17)	16	R/W	See section	54.6.115/ 2578
88E	Message Buffer Index Register (FR_MBIDXR17)	16	R/W	See section	54.6.116/ 2579
890	Message Buffer Configuration, Control, Status Register (FR_MBCCSR18)	16	R/W	0000h	54.6.113/ 2574
892	Message Buffer Cycle Counter Filter Register (FR_MBCCFR18)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
894	Message Buffer Frame ID Register (FR_MBFIDR18)	16	R/W	See section	54.6.115/ 2578
896	Message Buffer Index Register (FR_MBIDXR18)	16	R/W	See section	54.6.116/ 2579
898	Message Buffer Configuration, Control, Status Register (FR_MBCCSR19)	16	R/W	0000h	54.6.113/ 2574
89A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR19)	16	R/W	See section	54.6.114/ 2576
89C	Message Buffer Frame ID Register (FR_MBFIDR19)	16	R/W	See section	54.6.115/ 2578
89E	Message Buffer Index Register (FR_MBIDXR19)	16	R/W	See section	54.6.116/ 2579
8A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR20)	16	R/W	0000h	54.6.113/ 2574
8A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR20)	16	R/W	See section	54.6.114/ 2576
8A4	Message Buffer Frame ID Register (FR_MBFIDR20)	16	R/W	See section	54.6.115/ 2578
8A6	Message Buffer Index Register (FR_MBIDXR20)	16	R/W	See section	54.6.116/ 2579
8A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR21)	16	R/W	0000h	54.6.113/ 2574
8AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR21)	16	R/W	See section	54.6.114/ 2576
8AC	Message Buffer Frame ID Register (FR_MBFIDR21)	16	R/W	See section	54.6.115/ 2578
8AE	Message Buffer Index Register (FR_MBIDXR21)	16	R/W	See section	54.6.116/ 2579
8B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR22)	16	R/W	0000h	54.6.113/ 2574
8B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR22)	16	R/W	See section	54.6.114/ 2576
8B4	Message Buffer Frame ID Register (FR_MBFIDR22)	16	R/W	See section	54.6.115/ 2578
8B6	Message Buffer Index Register (FR_MBIDXR22)	16	R/W	See section	54.6.116/ 2579
8B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR23)	16	R/W	0000h	54.6.113/ 2574
8BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR23)	16	R/W	See section	54.6.114/ 2576
8BC	Message Buffer Frame ID Register (FR_MBFIDR23)	16	R/W	See section	54.6.115/ 2578
8BE	Message Buffer Index Register (FR_MBIDXR23)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR24)	16	R/W	0000h	54.6.113/ 2574
8C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR24)	16	R/W	See section	54.6.114/ 2576
8C4	Message Buffer Frame ID Register (FR_MBFIDR24)	16	R/W	See section	54.6.115/ 2578
8C6	Message Buffer Index Register (FR_MBIDXR24)	16	R/W	See section	54.6.116/ 2579
8C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR25)	16	R/W	0000h	54.6.113/ 2574
8CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR25)	16	R/W	See section	54.6.114/ 2576
8CC	Message Buffer Frame ID Register (FR_MBFIDR25)	16	R/W	See section	54.6.115/ 2578
8CE	Message Buffer Index Register (FR_MBIDXR25)	16	R/W	See section	54.6.116/ 2579
8D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR26)	16	R/W	0000h	54.6.113/ 2574
8D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR26)	16	R/W	See section	54.6.114/ 2576
8D4	Message Buffer Frame ID Register (FR_MBFIDR26)	16	R/W	See section	54.6.115/ 2578
8D6	Message Buffer Index Register (FR_MBIDXR26)	16	R/W	See section	54.6.116/ 2579
8D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR27)	16	R/W	0000h	54.6.113/ 2574
8DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR27)	16	R/W	See section	54.6.114/ 2576
8DC	Message Buffer Frame ID Register (FR_MBFIDR27)	16	R/W	See section	54.6.115/ 2578
8DE	Message Buffer Index Register (FR_MBIDXR27)	16	R/W	See section	54.6.116/ 2579
8E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR28)	16	R/W	0000h	54.6.113/ 2574
8E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR28)	16	R/W	See section	54.6.114/ 2576
8E4	Message Buffer Frame ID Register (FR_MBFIDR28)	16	R/W	See section	54.6.115/ 2578
8E6	Message Buffer Index Register (FR_MBIDXR28)	16	R/W	See section	54.6.116/ 2579
8E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR29)	16	R/W	0000h	54.6.113/ 2574
8EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR29)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8EC	Message Buffer Frame ID Register (FR_MBFIDR29)	16	R/W	See section	54.6.115/ 2578
8EE	Message Buffer Index Register (FR_MBIDXR29)	16	R/W	See section	54.6.116/ 2579
8F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR30)	16	R/W	0000h	54.6.113/ 2574
8F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR30)	16	R/W	See section	54.6.114/ 2576
8F4	Message Buffer Frame ID Register (FR_MBFIDR30)	16	R/W	See section	54.6.115/ 2578
8F6	Message Buffer Index Register (FR_MBIDXR30)	16	R/W	See section	54.6.116/ 2579
8F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR31)	16	R/W	0000h	54.6.113/ 2574
8FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR31)	16	R/W	See section	54.6.114/ 2576
8FC	Message Buffer Frame ID Register (FR_MBFIDR31)	16	R/W	See section	54.6.115/ 2578
8FE	Message Buffer Index Register (FR_MBIDXR31)	16	R/W	See section	54.6.116/ 2579
900	Message Buffer Configuration, Control, Status Register (FR_MBCCSR32)	16	R/W	0000h	54.6.113/ 2574
902	Message Buffer Cycle Counter Filter Register (FR_MBCCFR32)	16	R/W	See section	54.6.114/ 2576
904	Message Buffer Frame ID Register (FR_MBFIDR32)	16	R/W	See section	54.6.115/ 2578
906	Message Buffer Index Register (FR_MBIDXR32)	16	R/W	See section	54.6.116/ 2579
908	Message Buffer Configuration, Control, Status Register (FR_MBCCSR33)	16	R/W	0000h	54.6.113/ 2574
90A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR33)	16	R/W	See section	54.6.114/ 2576
90C	Message Buffer Frame ID Register (FR_MBFIDR33)	16	R/W	See section	54.6.115/ 2578
90E	Message Buffer Index Register (FR_MBIDXR33)	16	R/W	See section	54.6.116/ 2579
910	Message Buffer Configuration, Control, Status Register (FR_MBCCSR34)	16	R/W	0000h	54.6.113/ 2574
912	Message Buffer Cycle Counter Filter Register (FR_MBCCFR34)	16	R/W	See section	54.6.114/ 2576
914	Message Buffer Frame ID Register (FR_MBFIDR34)	16	R/W	See section	54.6.115/ 2578
916	Message Buffer Index Register (FR_MBIDXR34)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
918	Message Buffer Configuration, Control, Status Register (FR_MBCCSR35)	16	R/W	0000h	54.6.113/ 2574
91A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR35)	16	R/W	See section	54.6.114/ 2576
91C	Message Buffer Frame ID Register (FR_MBFIDR35)	16	R/W	See section	54.6.115/ 2578
91E	Message Buffer Index Register (FR_MBIDXR35)	16	R/W	See section	54.6.116/ 2579
920	Message Buffer Configuration, Control, Status Register (FR_MBCCSR36)	16	R/W	0000h	54.6.113/ 2574
922	Message Buffer Cycle Counter Filter Register (FR_MBCCFR36)	16	R/W	See section	54.6.114/ 2576
924	Message Buffer Frame ID Register (FR_MBFIDR36)	16	R/W	See section	54.6.115/ 2578
926	Message Buffer Index Register (FR_MBIDXR36)	16	R/W	See section	54.6.116/ 2579
928	Message Buffer Configuration, Control, Status Register (FR_MBCCSR37)	16	R/W	0000h	54.6.113/ 2574
92A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR37)	16	R/W	See section	54.6.114/ 2576
92C	Message Buffer Frame ID Register (FR_MBFIDR37)	16	R/W	See section	54.6.115/ 2578
92E	Message Buffer Index Register (FR_MBIDXR37)	16	R/W	See section	54.6.116/ 2579
930	Message Buffer Configuration, Control, Status Register (FR_MBCCSR38)	16	R/W	0000h	54.6.113/ 2574
932	Message Buffer Cycle Counter Filter Register (FR_MBCCFR38)	16	R/W	See section	54.6.114/ 2576
934	Message Buffer Frame ID Register (FR_MBFIDR38)	16	R/W	See section	54.6.115/ 2578
936	Message Buffer Index Register (FR_MBIDXR38)	16	R/W	See section	54.6.116/ 2579
938	Message Buffer Configuration, Control, Status Register (FR_MBCCSR39)	16	R/W	0000h	54.6.113/ 2574
93A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR39)	16	R/W	See section	54.6.114/ 2576
93C	Message Buffer Frame ID Register (FR_MBFIDR39)	16	R/W	See section	54.6.115/ 2578
93E	Message Buffer Index Register (FR_MBIDXR39)	16	R/W	See section	54.6.116/ 2579
940	Message Buffer Configuration, Control, Status Register (FR_MBCCSR40)	16	R/W	0000h	54.6.113/ 2574
942	Message Buffer Cycle Counter Filter Register (FR_MBCCFR40)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
944	Message Buffer Frame ID Register (FR_MBFIDR40)	16	R/W	See section	54.6.115/ 2578
946	Message Buffer Index Register (FR_MBIDXR40)	16	R/W	See section	54.6.116/ 2579
948	Message Buffer Configuration, Control, Status Register (FR_MBCCSR41)	16	R/W	0000h	54.6.113/ 2574
94A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR41)	16	R/W	See section	54.6.114/ 2576
94C	Message Buffer Frame ID Register (FR_MBFIDR41)	16	R/W	See section	54.6.115/ 2578
94E	Message Buffer Index Register (FR_MBIDXR41)	16	R/W	See section	54.6.116/ 2579
950	Message Buffer Configuration, Control, Status Register (FR_MBCCSR42)	16	R/W	0000h	54.6.113/ 2574
952	Message Buffer Cycle Counter Filter Register (FR_MBCCFR42)	16	R/W	See section	54.6.114/ 2576
954	Message Buffer Frame ID Register (FR_MBFIDR42)	16	R/W	See section	54.6.115/ 2578
956	Message Buffer Index Register (FR_MBIDXR42)	16	R/W	See section	54.6.116/ 2579
958	Message Buffer Configuration, Control, Status Register (FR_MBCCSR43)	16	R/W	0000h	54.6.113/ 2574
95A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR43)	16	R/W	See section	54.6.114/ 2576
95C	Message Buffer Frame ID Register (FR_MBFIDR43)	16	R/W	See section	54.6.115/ 2578
95E	Message Buffer Index Register (FR_MBIDXR43)	16	R/W	See section	54.6.116/ 2579
960	Message Buffer Configuration, Control, Status Register (FR_MBCCSR44)	16	R/W	0000h	54.6.113/ 2574
962	Message Buffer Cycle Counter Filter Register (FR_MBCCFR44)	16	R/W	See section	54.6.114/ 2576
964	Message Buffer Frame ID Register (FR_MBFIDR44)	16	R/W	See section	54.6.115/ 2578
966	Message Buffer Index Register (FR_MBIDXR44)	16	R/W	See section	54.6.116/ 2579
968	Message Buffer Configuration, Control, Status Register (FR_MBCCSR45)	16	R/W	0000h	54.6.113/ 2574
96A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR45)	16	R/W	See section	54.6.114/ 2576
96C	Message Buffer Frame ID Register (FR_MBFIDR45)	16	R/W	See section	54.6.115/ 2578
96E	Message Buffer Index Register (FR_MBIDXR45)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
970	Message Buffer Configuration, Control, Status Register (FR_MBCCSR46)	16	R/W	0000h	54.6.113/ 2574
972	Message Buffer Cycle Counter Filter Register (FR_MBCCFR46)	16	R/W	See section	54.6.114/ 2576
974	Message Buffer Frame ID Register (FR_MBFIDR46)	16	R/W	See section	54.6.115/ 2578
976	Message Buffer Index Register (FR_MBIDXR46)	16	R/W	See section	54.6.116/ 2579
978	Message Buffer Configuration, Control, Status Register (FR_MBCCSR47)	16	R/W	0000h	54.6.113/ 2574
97A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR47)	16	R/W	See section	54.6.114/ 2576
97C	Message Buffer Frame ID Register (FR_MBFIDR47)	16	R/W	See section	54.6.115/ 2578
97E	Message Buffer Index Register (FR_MBIDXR47)	16	R/W	See section	54.6.116/ 2579
980	Message Buffer Configuration, Control, Status Register (FR_MBCCSR48)	16	R/W	0000h	54.6.113/ 2574
982	Message Buffer Cycle Counter Filter Register (FR_MBCCFR48)	16	R/W	See section	54.6.114/ 2576
984	Message Buffer Frame ID Register (FR_MBFIDR48)	16	R/W	See section	54.6.115/ 2578
986	Message Buffer Index Register (FR_MBIDXR48)	16	R/W	See section	54.6.116/ 2579
988	Message Buffer Configuration, Control, Status Register (FR_MBCCSR49)	16	R/W	0000h	54.6.113/ 2574
98A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR49)	16	R/W	See section	54.6.114/ 2576
98C	Message Buffer Frame ID Register (FR_MBFIDR49)	16	R/W	See section	54.6.115/ 2578
98E	Message Buffer Index Register (FR_MBIDXR49)	16	R/W	See section	54.6.116/ 2579
990	Message Buffer Configuration, Control, Status Register (FR_MBCCSR50)	16	R/W	0000h	54.6.113/ 2574
992	Message Buffer Cycle Counter Filter Register (FR_MBCCFR50)	16	R/W	See section	54.6.114/ 2576
994	Message Buffer Frame ID Register (FR_MBFIDR50)	16	R/W	See section	54.6.115/ 2578
996	Message Buffer Index Register (FR_MBIDXR50)	16	R/W	See section	54.6.116/ 2579
998	Message Buffer Configuration, Control, Status Register (FR_MBCCSR51)	16	R/W	0000h	54.6.113/ 2574
99A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR51)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
99C	Message Buffer Frame ID Register (FR_MBFIDR51)	16	R/W	See section	54.6.115/ 2578
99E	Message Buffer Index Register (FR_MBIDXR51)	16	R/W	See section	54.6.116/ 2579
9A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR52)	16	R/W	0000h	54.6.113/ 2574
9A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR52)	16	R/W	See section	54.6.114/ 2576
9A4	Message Buffer Frame ID Register (FR_MBFIDR52)	16	R/W	See section	54.6.115/ 2578
9A6	Message Buffer Index Register (FR_MBIDXR52)	16	R/W	See section	54.6.116/ 2579
9A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR53)	16	R/W	0000h	54.6.113/ 2574
9AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR53)	16	R/W	See section	54.6.114/ 2576
9AC	Message Buffer Frame ID Register (FR_MBFIDR53)	16	R/W	See section	54.6.115/ 2578
9AE	Message Buffer Index Register (FR_MBIDXR53)	16	R/W	See section	54.6.116/ 2579
9B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR54)	16	R/W	0000h	54.6.113/ 2574
9B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR54)	16	R/W	See section	54.6.114/ 2576
9B4	Message Buffer Frame ID Register (FR_MBFIDR54)	16	R/W	See section	54.6.115/ 2578
9B6	Message Buffer Index Register (FR_MBIDXR54)	16	R/W	See section	54.6.116/ 2579
9B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR55)	16	R/W	0000h	54.6.113/ 2574
9BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR55)	16	R/W	See section	54.6.114/ 2576
9BC	Message Buffer Frame ID Register (FR_MBFIDR55)	16	R/W	See section	54.6.115/ 2578
9BE	Message Buffer Index Register (FR_MBIDXR55)	16	R/W	See section	54.6.116/ 2579
9C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR56)	16	R/W	0000h	54.6.113/ 2574
9C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR56)	16	R/W	See section	54.6.114/ 2576
9C4	Message Buffer Frame ID Register (FR_MBFIDR56)	16	R/W	See section	54.6.115/ 2578
9C6	Message Buffer Index Register (FR_MBIDXR56)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR57)	16	R/W	0000h	54.6.113/ 2574
9CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR57)	16	R/W	See section	54.6.114/ 2576
9CC	Message Buffer Frame ID Register (FR_MBFIDR57)	16	R/W	See section	54.6.115/ 2578
9CE	Message Buffer Index Register (FR_MBIDXR57)	16	R/W	See section	54.6.116/ 2579
9D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR58)	16	R/W	0000h	54.6.113/ 2574
9D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR58)	16	R/W	See section	54.6.114/ 2576
9D4	Message Buffer Frame ID Register (FR_MBFIDR58)	16	R/W	See section	54.6.115/ 2578
9D6	Message Buffer Index Register (FR_MBIDXR58)	16	R/W	See section	54.6.116/ 2579
9D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR59)	16	R/W	0000h	54.6.113/ 2574
9DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR59)	16	R/W	See section	54.6.114/ 2576
9DC	Message Buffer Frame ID Register (FR_MBFIDR59)	16	R/W	See section	54.6.115/ 2578
9DE	Message Buffer Index Register (FR_MBIDXR59)	16	R/W	See section	54.6.116/ 2579
9E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR60)	16	R/W	0000h	54.6.113/ 2574
9E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR60)	16	R/W	See section	54.6.114/ 2576
9E4	Message Buffer Frame ID Register (FR_MBFIDR60)	16	R/W	See section	54.6.115/ 2578
9E6	Message Buffer Index Register (FR_MBIDXR60)	16	R/W	See section	54.6.116/ 2579
9E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR61)	16	R/W	0000h	54.6.113/ 2574
9EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR61)	16	R/W	See section	54.6.114/ 2576
9EC	Message Buffer Frame ID Register (FR_MBFIDR61)	16	R/W	See section	54.6.115/ 2578
9EE	Message Buffer Index Register (FR_MBIDXR61)	16	R/W	See section	54.6.116/ 2579
9F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR62)	16	R/W	0000h	54.6.113/ 2574
9F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR62)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9F4	Message Buffer Frame ID Register (FR_MBFIDR62)	16	R/W	See section	54.6.115/ 2578
9F6	Message Buffer Index Register (FR_MBIDXR62)	16	R/W	See section	54.6.116/ 2579
9F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR63)	16	R/W	0000h	54.6.113/ 2574
9FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR63)	16	R/W	See section	54.6.114/ 2576
9FC	Message Buffer Frame ID Register (FR_MBFIDR63)	16	R/W	See section	54.6.115/ 2578
9FE	Message Buffer Index Register (FR_MBIDXR63)	16	R/W	See section	54.6.116/ 2579
A00	Message Buffer Configuration, Control, Status Register (FR_MBCCSR64)	16	R/W	0000h	54.6.113/ 2574
A02	Message Buffer Cycle Counter Filter Register (FR_MBCCFR64)	16	R/W	See section	54.6.114/ 2576
A04	Message Buffer Frame ID Register (FR_MBFIDR64)	16	R/W	See section	54.6.115/ 2578
A06	Message Buffer Index Register (FR_MBIDXR64)	16	R/W	See section	54.6.116/ 2579
A08	Message Buffer Configuration, Control, Status Register (FR_MBCCSR65)	16	R/W	0000h	54.6.113/ 2574
A0A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR65)	16	R/W	See section	54.6.114/ 2576
A0C	Message Buffer Frame ID Register (FR_MBFIDR65)	16	R/W	See section	54.6.115/ 2578
A0E	Message Buffer Index Register (FR_MBIDXR65)	16	R/W	See section	54.6.116/ 2579
A10	Message Buffer Configuration, Control, Status Register (FR_MBCCSR66)	16	R/W	0000h	54.6.113/ 2574
A12	Message Buffer Cycle Counter Filter Register (FR_MBCCFR66)	16	R/W	See section	54.6.114/ 2576
A14	Message Buffer Frame ID Register (FR_MBFIDR66)	16	R/W	See section	54.6.115/ 2578
A16	Message Buffer Index Register (FR_MBIDXR66)	16	R/W	See section	54.6.116/ 2579
A18	Message Buffer Configuration, Control, Status Register (FR_MBCCSR67)	16	R/W	0000h	54.6.113/ 2574
A1A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR67)	16	R/W	See section	54.6.114/ 2576
A1C	Message Buffer Frame ID Register (FR_MBFIDR67)	16	R/W	See section	54.6.115/ 2578
A1E	Message Buffer Index Register (FR_MBIDXR67)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A20	Message Buffer Configuration, Control, Status Register (FR_MBCCSR68)	16	R/W	0000h	54.6.113/ 2574
A22	Message Buffer Cycle Counter Filter Register (FR_MBCCFR68)	16	R/W	See section	54.6.114/ 2576
A24	Message Buffer Frame ID Register (FR_MBFIDR68)	16	R/W	See section	54.6.115/ 2578
A26	Message Buffer Index Register (FR_MBIDXR68)	16	R/W	See section	54.6.116/ 2579
A28	Message Buffer Configuration, Control, Status Register (FR_MBCCSR69)	16	R/W	0000h	54.6.113/ 2574
A2A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR69)	16	R/W	See section	54.6.114/ 2576
A2C	Message Buffer Frame ID Register (FR_MBFIDR69)	16	R/W	See section	54.6.115/ 2578
A2E	Message Buffer Index Register (FR_MBIDXR69)	16	R/W	See section	54.6.116/ 2579
A30	Message Buffer Configuration, Control, Status Register (FR_MBCCSR70)	16	R/W	0000h	54.6.113/ 2574
A32	Message Buffer Cycle Counter Filter Register (FR_MBCCFR70)	16	R/W	See section	54.6.114/ 2576
A34	Message Buffer Frame ID Register (FR_MBFIDR70)	16	R/W	See section	54.6.115/ 2578
A36	Message Buffer Index Register (FR_MBIDXR70)	16	R/W	See section	54.6.116/ 2579
A38	Message Buffer Configuration, Control, Status Register (FR_MBCCSR71)	16	R/W	0000h	54.6.113/ 2574
A3A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR71)	16	R/W	See section	54.6.114/ 2576
A3C	Message Buffer Frame ID Register (FR_MBFIDR71)	16	R/W	See section	54.6.115/ 2578
A3E	Message Buffer Index Register (FR_MBIDXR71)	16	R/W	See section	54.6.116/ 2579
A40	Message Buffer Configuration, Control, Status Register (FR_MBCCSR72)	16	R/W	0000h	54.6.113/ 2574
A42	Message Buffer Cycle Counter Filter Register (FR_MBCCFR72)	16	R/W	See section	54.6.114/ 2576
A44	Message Buffer Frame ID Register (FR_MBFIDR72)	16	R/W	See section	54.6.115/ 2578
A46	Message Buffer Index Register (FR_MBIDXR72)	16	R/W	See section	54.6.116/ 2579
A48	Message Buffer Configuration, Control, Status Register (FR_MBCCSR73)	16	R/W	0000h	54.6.113/ 2574
A4A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR73)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4C	Message Buffer Frame ID Register (FR_MBFIDR73)	16	R/W	See section	54.6.115/ 2578
A4E	Message Buffer Index Register (FR_MBIDXR73)	16	R/W	See section	54.6.116/ 2579
A50	Message Buffer Configuration, Control, Status Register (FR_MBCCSR74)	16	R/W	0000h	54.6.113/ 2574
A52	Message Buffer Cycle Counter Filter Register (FR_MBCCFR74)	16	R/W	See section	54.6.114/ 2576
A54	Message Buffer Frame ID Register (FR_MBFIDR74)	16	R/W	See section	54.6.115/ 2578
A56	Message Buffer Index Register (FR_MBIDXR74)	16	R/W	See section	54.6.116/ 2579
A58	Message Buffer Configuration, Control, Status Register (FR_MBCCSR75)	16	R/W	0000h	54.6.113/ 2574
A5A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR75)	16	R/W	See section	54.6.114/ 2576
A5C	Message Buffer Frame ID Register (FR_MBFIDR75)	16	R/W	See section	54.6.115/ 2578
A5E	Message Buffer Index Register (FR_MBIDXR75)	16	R/W	See section	54.6.116/ 2579
A60	Message Buffer Configuration, Control, Status Register (FR_MBCCSR76)	16	R/W	0000h	54.6.113/ 2574
A62	Message Buffer Cycle Counter Filter Register (FR_MBCCFR76)	16	R/W	See section	54.6.114/ 2576
A64	Message Buffer Frame ID Register (FR_MBFIDR76)	16	R/W	See section	54.6.115/ 2578
A66	Message Buffer Index Register (FR_MBIDXR76)	16	R/W	See section	54.6.116/ 2579
A68	Message Buffer Configuration, Control, Status Register (FR_MBCCSR77)	16	R/W	0000h	54.6.113/ 2574
A6A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR77)	16	R/W	See section	54.6.114/ 2576
A6C	Message Buffer Frame ID Register (FR_MBFIDR77)	16	R/W	See section	54.6.115/ 2578
A6E	Message Buffer Index Register (FR_MBIDXR77)	16	R/W	See section	54.6.116/ 2579
A70	Message Buffer Configuration, Control, Status Register (FR_MBCCSR78)	16	R/W	0000h	54.6.113/ 2574
A72	Message Buffer Cycle Counter Filter Register (FR_MBCCFR78)	16	R/W	See section	54.6.114/ 2576
A74	Message Buffer Frame ID Register (FR_MBFIDR78)	16	R/W	See section	54.6.115/ 2578
A76	Message Buffer Index Register (FR_MBIDXR78)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A78	Message Buffer Configuration, Control, Status Register (FR_MBCCSR79)	16	R/W	0000h	54.6.113/ 2574
A7A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR79)	16	R/W	See section	54.6.114/ 2576
A7C	Message Buffer Frame ID Register (FR_MBFIDR79)	16	R/W	See section	54.6.115/ 2578
A7E	Message Buffer Index Register (FR_MBIDXR79)	16	R/W	See section	54.6.116/ 2579
A80	Message Buffer Configuration, Control, Status Register (FR_MBCCSR80)	16	R/W	0000h	54.6.113/ 2574
A82	Message Buffer Cycle Counter Filter Register (FR_MBCCFR80)	16	R/W	See section	54.6.114/ 2576
A84	Message Buffer Frame ID Register (FR_MBFIDR80)	16	R/W	See section	54.6.115/ 2578
A86	Message Buffer Index Register (FR_MBIDXR80)	16	R/W	See section	54.6.116/ 2579
A88	Message Buffer Configuration, Control, Status Register (FR_MBCCSR81)	16	R/W	0000h	54.6.113/ 2574
A8A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR81)	16	R/W	See section	54.6.114/ 2576
A8C	Message Buffer Frame ID Register (FR_MBFIDR81)	16	R/W	See section	54.6.115/ 2578
A8E	Message Buffer Index Register (FR_MBIDXR81)	16	R/W	See section	54.6.116/ 2579
A90	Message Buffer Configuration, Control, Status Register (FR_MBCCSR82)	16	R/W	0000h	54.6.113/ 2574
A92	Message Buffer Cycle Counter Filter Register (FR_MBCCFR82)	16	R/W	See section	54.6.114/ 2576
A94	Message Buffer Frame ID Register (FR_MBFIDR82)	16	R/W	See section	54.6.115/ 2578
A96	Message Buffer Index Register (FR_MBIDXR82)	16	R/W	See section	54.6.116/ 2579
A98	Message Buffer Configuration, Control, Status Register (FR_MBCCSR83)	16	R/W	0000h	54.6.113/ 2574
A9A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR83)	16	R/W	See section	54.6.114/ 2576
A9C	Message Buffer Frame ID Register (FR_MBFIDR83)	16	R/W	See section	54.6.115/ 2578
A9E	Message Buffer Index Register (FR_MBIDXR83)	16	R/W	See section	54.6.116/ 2579
AA0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR84)	16	R/W	0000h	54.6.113/ 2574
AA2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR84)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AA4	Message Buffer Frame ID Register (FR_MBFIDR84)	16	R/W	See section	54.6.115/ 2578
AA6	Message Buffer Index Register (FR_MBIDXR84)	16	R/W	See section	54.6.116/ 2579
AA8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR85)	16	R/W	0000h	54.6.113/ 2574
AAA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR85)	16	R/W	See section	54.6.114/ 2576
AAC	Message Buffer Frame ID Register (FR_MBFIDR85)	16	R/W	See section	54.6.115/ 2578
AAE	Message Buffer Index Register (FR_MBIDXR85)	16	R/W	See section	54.6.116/ 2579
AB0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR86)	16	R/W	0000h	54.6.113/ 2574
AB2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR86)	16	R/W	See section	54.6.114/ 2576
AB4	Message Buffer Frame ID Register (FR_MBFIDR86)	16	R/W	See section	54.6.115/ 2578
AB6	Message Buffer Index Register (FR_MBIDXR86)	16	R/W	See section	54.6.116/ 2579
AB8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR87)	16	R/W	0000h	54.6.113/ 2574
ABA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR87)	16	R/W	See section	54.6.114/ 2576
ABC	Message Buffer Frame ID Register (FR_MBFIDR87)	16	R/W	See section	54.6.115/ 2578
ABE	Message Buffer Index Register (FR_MBIDXR87)	16	R/W	See section	54.6.116/ 2579
AC0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR88)	16	R/W	0000h	54.6.113/ 2574
AC2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR88)	16	R/W	See section	54.6.114/ 2576
AC4	Message Buffer Frame ID Register (FR_MBFIDR88)	16	R/W	See section	54.6.115/ 2578
AC6	Message Buffer Index Register (FR_MBIDXR88)	16	R/W	See section	54.6.116/ 2579
AC8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR89)	16	R/W	0000h	54.6.113/ 2574
ACA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR89)	16	R/W	See section	54.6.114/ 2576
ACC	Message Buffer Frame ID Register (FR_MBFIDR89)	16	R/W	See section	54.6.115/ 2578
ACE	Message Buffer Index Register (FR_MBIDXR89)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AD0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR90)	16	R/W	0000h	54.6.113/ 2574
AD2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR90)	16	R/W	See section	54.6.114/ 2576
AD4	Message Buffer Frame ID Register (FR_MBFIDR90)	16	R/W	See section	54.6.115/ 2578
AD6	Message Buffer Index Register (FR_MBIDXR90)	16	R/W	See section	54.6.116/ 2579
AD8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR91)	16	R/W	0000h	54.6.113/ 2574
ADA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR91)	16	R/W	See section	54.6.114/ 2576
ADC	Message Buffer Frame ID Register (FR_MBFIDR91)	16	R/W	See section	54.6.115/ 2578
ADE	Message Buffer Index Register (FR_MBIDXR91)	16	R/W	See section	54.6.116/ 2579
AE0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR92)	16	R/W	0000h	54.6.113/ 2574
AE2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR92)	16	R/W	See section	54.6.114/ 2576
AE4	Message Buffer Frame ID Register (FR_MBFIDR92)	16	R/W	See section	54.6.115/ 2578
AE6	Message Buffer Index Register (FR_MBIDXR92)	16	R/W	See section	54.6.116/ 2579
AE8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR93)	16	R/W	0000h	54.6.113/ 2574
AEA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR93)	16	R/W	See section	54.6.114/ 2576
AEC	Message Buffer Frame ID Register (FR_MBFIDR93)	16	R/W	See section	54.6.115/ 2578
AEE	Message Buffer Index Register (FR_MBIDXR93)	16	R/W	See section	54.6.116/ 2579
AF0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR94)	16	R/W	0000h	54.6.113/ 2574
AF2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR94)	16	R/W	See section	54.6.114/ 2576
AF4	Message Buffer Frame ID Register (FR_MBFIDR94)	16	R/W	See section	54.6.115/ 2578
AF6	Message Buffer Index Register (FR_MBIDXR94)	16	R/W	See section	54.6.116/ 2579
AF8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR95)	16	R/W	0000h	54.6.113/ 2574
AFA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR95)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AFC	Message Buffer Frame ID Register (FR_MBFIDR95)	16	R/W	See section	54.6.115/ 2578
AFE	Message Buffer Index Register (FR_MBIDXR95)	16	R/W	See section	54.6.116/ 2579
B00	Message Buffer Configuration, Control, Status Register (FR_MBCCSR96)	16	R/W	0000h	54.6.113/ 2574
B02	Message Buffer Cycle Counter Filter Register (FR_MBCCFR96)	16	R/W	See section	54.6.114/ 2576
B04	Message Buffer Frame ID Register (FR_MBFIDR96)	16	R/W	See section	54.6.115/ 2578
B06	Message Buffer Index Register (FR_MBIDXR96)	16	R/W	See section	54.6.116/ 2579
B08	Message Buffer Configuration, Control, Status Register (FR_MBCCSR97)	16	R/W	0000h	54.6.113/ 2574
B0A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR97)	16	R/W	See section	54.6.114/ 2576
B0C	Message Buffer Frame ID Register (FR_MBFIDR97)	16	R/W	See section	54.6.115/ 2578
B0E	Message Buffer Index Register (FR_MBIDXR97)	16	R/W	See section	54.6.116/ 2579
B10	Message Buffer Configuration, Control, Status Register (FR_MBCCSR98)	16	R/W	0000h	54.6.113/ 2574
B12	Message Buffer Cycle Counter Filter Register (FR_MBCCFR98)	16	R/W	See section	54.6.114/ 2576
B14	Message Buffer Frame ID Register (FR_MBFIDR98)	16	R/W	See section	54.6.115/ 2578
B16	Message Buffer Index Register (FR_MBIDXR98)	16	R/W	See section	54.6.116/ 2579
B18	Message Buffer Configuration, Control, Status Register (FR_MBCCSR99)	16	R/W	0000h	54.6.113/ 2574
B1A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR99)	16	R/W	See section	54.6.114/ 2576
B1C	Message Buffer Frame ID Register (FR_MBFIDR99)	16	R/W	See section	54.6.115/ 2578
B1E	Message Buffer Index Register (FR_MBIDXR99)	16	R/W	See section	54.6.116/ 2579
B20	Message Buffer Configuration, Control, Status Register (FR_MBCCSR100)	16	R/W	0000h	54.6.113/ 2574
B22	Message Buffer Cycle Counter Filter Register (FR_MBCCFR100)	16	R/W	See section	54.6.114/ 2576
B24	Message Buffer Frame ID Register (FR_MBFIDR100)	16	R/W	See section	54.6.115/ 2578
B26	Message Buffer Index Register (FR_MBIDXR100)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B28	Message Buffer Configuration, Control, Status Register (FR_MBCCSR101)	16	R/W	0000h	54.6.113/ 2574
B2A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR101)	16	R/W	See section	54.6.114/ 2576
B2C	Message Buffer Frame ID Register (FR_MBFIDR101)	16	R/W	See section	54.6.115/ 2578
B2E	Message Buffer Index Register (FR_MBIDXR101)	16	R/W	See section	54.6.116/ 2579
B30	Message Buffer Configuration, Control, Status Register (FR_MBCCSR102)	16	R/W	0000h	54.6.113/ 2574
B32	Message Buffer Cycle Counter Filter Register (FR_MBCCFR102)	16	R/W	See section	54.6.114/ 2576
B34	Message Buffer Frame ID Register (FR_MBFIDR102)	16	R/W	See section	54.6.115/ 2578
B36	Message Buffer Index Register (FR_MBIDXR102)	16	R/W	See section	54.6.116/ 2579
B38	Message Buffer Configuration, Control, Status Register (FR_MBCCSR103)	16	R/W	0000h	54.6.113/ 2574
B3A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR103)	16	R/W	See section	54.6.114/ 2576
B3C	Message Buffer Frame ID Register (FR_MBFIDR103)	16	R/W	See section	54.6.115/ 2578
B3E	Message Buffer Index Register (FR_MBIDXR103)	16	R/W	See section	54.6.116/ 2579
B40	Message Buffer Configuration, Control, Status Register (FR_MBCCSR104)	16	R/W	0000h	54.6.113/ 2574
B42	Message Buffer Cycle Counter Filter Register (FR_MBCCFR104)	16	R/W	See section	54.6.114/ 2576
B44	Message Buffer Frame ID Register (FR_MBFIDR104)	16	R/W	See section	54.6.115/ 2578
B46	Message Buffer Index Register (FR_MBIDXR104)	16	R/W	See section	54.6.116/ 2579
B48	Message Buffer Configuration, Control, Status Register (FR_MBCCSR105)	16	R/W	0000h	54.6.113/ 2574
B4A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR105)	16	R/W	See section	54.6.114/ 2576
B4C	Message Buffer Frame ID Register (FR_MBFIDR105)	16	R/W	See section	54.6.115/ 2578
B4E	Message Buffer Index Register (FR_MBIDXR105)	16	R/W	See section	54.6.116/ 2579
B50	Message Buffer Configuration, Control, Status Register (FR_MBCCSR106)	16	R/W	0000h	54.6.113/ 2574
B52	Message Buffer Cycle Counter Filter Register (FR_MBCCFR106)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B54	Message Buffer Frame ID Register (FR_MBFIDR106)	16	R/W	See section	54.6.115/ 2578
B56	Message Buffer Index Register (FR_MBIDXR106)	16	R/W	See section	54.6.116/ 2579
B58	Message Buffer Configuration, Control, Status Register (FR_MBCCSR107)	16	R/W	0000h	54.6.113/ 2574
B5A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR107)	16	R/W	See section	54.6.114/ 2576
B5C	Message Buffer Frame ID Register (FR_MBFIDR107)	16	R/W	See section	54.6.115/ 2578
B5E	Message Buffer Index Register (FR_MBIDXR107)	16	R/W	See section	54.6.116/ 2579
B60	Message Buffer Configuration, Control, Status Register (FR_MBCCSR108)	16	R/W	0000h	54.6.113/ 2574
B62	Message Buffer Cycle Counter Filter Register (FR_MBCCFR108)	16	R/W	See section	54.6.114/ 2576
B64	Message Buffer Frame ID Register (FR_MBFIDR108)	16	R/W	See section	54.6.115/ 2578
B66	Message Buffer Index Register (FR_MBIDXR108)	16	R/W	See section	54.6.116/ 2579
B68	Message Buffer Configuration, Control, Status Register (FR_MBCCSR109)	16	R/W	0000h	54.6.113/ 2574
B6A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR109)	16	R/W	See section	54.6.114/ 2576
B6C	Message Buffer Frame ID Register (FR_MBFIDR109)	16	R/W	See section	54.6.115/ 2578
B6E	Message Buffer Index Register (FR_MBIDXR109)	16	R/W	See section	54.6.116/ 2579
B70	Message Buffer Configuration, Control, Status Register (FR_MBCCSR110)	16	R/W	0000h	54.6.113/ 2574
B72	Message Buffer Cycle Counter Filter Register (FR_MBCCFR110)	16	R/W	See section	54.6.114/ 2576
B74	Message Buffer Frame ID Register (FR_MBFIDR110)	16	R/W	See section	54.6.115/ 2578
B76	Message Buffer Index Register (FR_MBIDXR110)	16	R/W	See section	54.6.116/ 2579
B78	Message Buffer Configuration, Control, Status Register (FR_MBCCSR111)	16	R/W	0000h	54.6.113/ 2574
B7A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR111)	16	R/W	See section	54.6.114/ 2576
B7C	Message Buffer Frame ID Register (FR_MBFIDR111)	16	R/W	See section	54.6.115/ 2578
B7E	Message Buffer Index Register (FR_MBIDXR111)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B80	Message Buffer Configuration, Control, Status Register (FR_MBCCSR112)	16	R/W	0000h	54.6.113/ 2574
B82	Message Buffer Cycle Counter Filter Register (FR_MBCCFR112)	16	R/W	See section	54.6.114/ 2576
B84	Message Buffer Frame ID Register (FR_MBFIDR112)	16	R/W	See section	54.6.115/ 2578
B86	Message Buffer Index Register (FR_MBIDXR112)	16	R/W	See section	54.6.116/ 2579
B88	Message Buffer Configuration, Control, Status Register (FR_MBCCSR113)	16	R/W	0000h	54.6.113/ 2574
B8A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR113)	16	R/W	See section	54.6.114/ 2576
B8C	Message Buffer Frame ID Register (FR_MBFIDR113)	16	R/W	See section	54.6.115/ 2578
B8E	Message Buffer Index Register (FR_MBIDXR113)	16	R/W	See section	54.6.116/ 2579
B90	Message Buffer Configuration, Control, Status Register (FR_MBCCSR114)	16	R/W	0000h	54.6.113/ 2574
B92	Message Buffer Cycle Counter Filter Register (FR_MBCCFR114)	16	R/W	See section	54.6.114/ 2576
B94	Message Buffer Frame ID Register (FR_MBFIDR114)	16	R/W	See section	54.6.115/ 2578
B96	Message Buffer Index Register (FR_MBIDXR114)	16	R/W	See section	54.6.116/ 2579
B98	Message Buffer Configuration, Control, Status Register (FR_MBCCSR115)	16	R/W	0000h	54.6.113/ 2574
B9A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR115)	16	R/W	See section	54.6.114/ 2576
B9C	Message Buffer Frame ID Register (FR_MBFIDR115)	16	R/W	See section	54.6.115/ 2578
B9E	Message Buffer Index Register (FR_MBIDXR115)	16	R/W	See section	54.6.116/ 2579
BA0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR116)	16	R/W	0000h	54.6.113/ 2574
BA2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR116)	16	R/W	See section	54.6.114/ 2576
BA4	Message Buffer Frame ID Register (FR_MBFIDR116)	16	R/W	See section	54.6.115/ 2578
BA6	Message Buffer Index Register (FR_MBIDXR116)	16	R/W	See section	54.6.116/ 2579
BA8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR117)	16	R/W	0000h	54.6.113/ 2574
BAA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR117)	16	R/W	See section	54.6.114/ 2576

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BAC	Message Buffer Frame ID Register (FR_MBFIDR117)	16	R/W	See section	54.6.115/ 2578
BAE	Message Buffer Index Register (FR_MBIDXR117)	16	R/W	See section	54.6.116/ 2579
BB0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR118)	16	R/W	0000h	54.6.113/ 2574
BB2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR118)	16	R/W	See section	54.6.114/ 2576
BB4	Message Buffer Frame ID Register (FR_MBFIDR118)	16	R/W	See section	54.6.115/ 2578
BB6	Message Buffer Index Register (FR_MBIDXR118)	16	R/W	See section	54.6.116/ 2579
BB8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR119)	16	R/W	0000h	54.6.113/ 2574
BBA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR119)	16	R/W	See section	54.6.114/ 2576
BBC	Message Buffer Frame ID Register (FR_MBFIDR119)	16	R/W	See section	54.6.115/ 2578
BBE	Message Buffer Index Register (FR_MBIDXR119)	16	R/W	See section	54.6.116/ 2579
BC0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR120)	16	R/W	0000h	54.6.113/ 2574
BC2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR120)	16	R/W	See section	54.6.114/ 2576
BC4	Message Buffer Frame ID Register (FR_MBFIDR120)	16	R/W	See section	54.6.115/ 2578
BC6	Message Buffer Index Register (FR_MBIDXR120)	16	R/W	See section	54.6.116/ 2579
BC8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR121)	16	R/W	0000h	54.6.113/ 2574
BCA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR121)	16	R/W	See section	54.6.114/ 2576
BCC	Message Buffer Frame ID Register (FR_MBFIDR121)	16	R/W	See section	54.6.115/ 2578
BCE	Message Buffer Index Register (FR_MBIDXR121)	16	R/W	See section	54.6.116/ 2579
BD0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR122)	16	R/W	0000h	54.6.113/ 2574
BD2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR122)	16	R/W	See section	54.6.114/ 2576
BD4	Message Buffer Frame ID Register (FR_MBFIDR122)	16	R/W	See section	54.6.115/ 2578
BD6	Message Buffer Index Register (FR_MBIDXR122)	16	R/W	See section	54.6.116/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BD8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR123)	16	R/W	0000h	54.6.113/ 2574
BDA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR123)	16	R/W	See section	54.6.114/ 2576
BDC	Message Buffer Frame ID Register (FR_MBFIDR123)	16	R/W	See section	54.6.115/ 2578
BDE	Message Buffer Index Register (FR_MBIDXR123)	16	R/W	See section	54.6.116/ 2579
BE0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR124)	16	R/W	0000h	54.6.113/ 2574
BE2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR124)	16	R/W	See section	54.6.114/ 2576
BE4	Message Buffer Frame ID Register (FR_MBFIDR124)	16	R/W	See section	54.6.115/ 2578
BE6	Message Buffer Index Register (FR_MBIDXR124)	16	R/W	See section	54.6.116/ 2579
BE8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR125)	16	R/W	0000h	54.6.113/ 2574
BEA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR125)	16	R/W	See section	54.6.114/ 2576
BEC	Message Buffer Frame ID Register (FR_MBFIDR125)	16	R/W	See section	54.6.115/ 2578
BEE	Message Buffer Index Register (FR_MBIDXR125)	16	R/W	See section	54.6.116/ 2579
BF0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR126)	16	R/W	0000h	54.6.113/ 2574
BF2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR126)	16	R/W	See section	54.6.114/ 2576
BF4	Message Buffer Frame ID Register (FR_MBFIDR126)	16	R/W	See section	54.6.115/ 2578
BF6	Message Buffer Index Register (FR_MBIDXR126)	16	R/W	See section	54.6.116/ 2579
BF8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR127)	16	R/W	0000h	54.6.113/ 2574
BFA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR127)	16	R/W	See section	54.6.114/ 2576
BFC	Message Buffer Frame ID Register (FR_MBFIDR127)	16	R/W	See section	54.6.115/ 2578
BFE	Message Buffer Index Register (FR_MBIDXR127)	16	R/W	See section	54.6.116/ 2579
1000	Message Buffer Data Field Offset Register (FR_MBDOR0)	16	R/W	See section	54.6.117/ 2579
1002	Message Buffer Data Field Offset Register (FR_MBDOR1)	16	R/W	See section	54.6.117/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1004	Message Buffer Data Field Offset Register (FR_MBDOR2)	16	R/W	See section	54.6.117/ 2579
1006	Message Buffer Data Field Offset Register (FR_MBDOR3)	16	R/W	See section	54.6.117/ 2579
1008	Message Buffer Data Field Offset Register (FR_MBDOR4)	16	R/W	See section	54.6.117/ 2579
100A	Message Buffer Data Field Offset Register (FR_MBDOR5)	16	R/W	See section	54.6.117/ 2579
100C	Message Buffer Data Field Offset Register (FR_MBDOR6)	16	R/W	See section	54.6.117/ 2579
100E	Message Buffer Data Field Offset Register (FR_MBDOR7)	16	R/W	See section	54.6.117/ 2579
1010	Message Buffer Data Field Offset Register (FR_MBDOR8)	16	R/W	See section	54.6.117/ 2579
1012	Message Buffer Data Field Offset Register (FR_MBDOR9)	16	R/W	See section	54.6.117/ 2579
1014	Message Buffer Data Field Offset Register (FR_MBDOR10)	16	R/W	See section	54.6.117/ 2579
1016	Message Buffer Data Field Offset Register (FR_MBDOR11)	16	R/W	See section	54.6.117/ 2579
1018	Message Buffer Data Field Offset Register (FR_MBDOR12)	16	R/W	See section	54.6.117/ 2579
101A	Message Buffer Data Field Offset Register (FR_MBDOR13)	16	R/W	See section	54.6.117/ 2579
101C	Message Buffer Data Field Offset Register (FR_MBDOR14)	16	R/W	See section	54.6.117/ 2579
101E	Message Buffer Data Field Offset Register (FR_MBDOR15)	16	R/W	See section	54.6.117/ 2579
1020	Message Buffer Data Field Offset Register (FR_MBDOR16)	16	R/W	See section	54.6.117/ 2579
1022	Message Buffer Data Field Offset Register (FR_MBDOR17)	16	R/W	See section	54.6.117/ 2579
1024	Message Buffer Data Field Offset Register (FR_MBDOR18)	16	R/W	See section	54.6.117/ 2579
1026	Message Buffer Data Field Offset Register (FR_MBDOR19)	16	R/W	See section	54.6.117/ 2579
1028	Message Buffer Data Field Offset Register (FR_MBDOR20)	16	R/W	See section	54.6.117/ 2579
102A	Message Buffer Data Field Offset Register (FR_MBDOR21)	16	R/W	See section	54.6.117/ 2579
102C	Message Buffer Data Field Offset Register (FR_MBDOR22)	16	R/W	See section	54.6.117/ 2579
102E	Message Buffer Data Field Offset Register (FR_MBDOR23)	16	R/W	See section	54.6.117/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1030	Message Buffer Data Field Offset Register (FR_MBDOR24)	16	R/W	See section	54.6.117/ 2579
1032	Message Buffer Data Field Offset Register (FR_MBDOR25)	16	R/W	See section	54.6.117/ 2579
1034	Message Buffer Data Field Offset Register (FR_MBDOR26)	16	R/W	See section	54.6.117/ 2579
1036	Message Buffer Data Field Offset Register (FR_MBDOR27)	16	R/W	See section	54.6.117/ 2579
1038	Message Buffer Data Field Offset Register (FR_MBDOR28)	16	R/W	See section	54.6.117/ 2579
103A	Message Buffer Data Field Offset Register (FR_MBDOR29)	16	R/W	See section	54.6.117/ 2579
103C	Message Buffer Data Field Offset Register (FR_MBDOR30)	16	R/W	See section	54.6.117/ 2579
103E	Message Buffer Data Field Offset Register (FR_MBDOR31)	16	R/W	See section	54.6.117/ 2579
1040	Message Buffer Data Field Offset Register (FR_MBDOR32)	16	R/W	See section	54.6.117/ 2579
1042	Message Buffer Data Field Offset Register (FR_MBDOR33)	16	R/W	See section	54.6.117/ 2579
1044	Message Buffer Data Field Offset Register (FR_MBDOR34)	16	R/W	See section	54.6.117/ 2579
1046	Message Buffer Data Field Offset Register (FR_MBDOR35)	16	R/W	See section	54.6.117/ 2579
1048	Message Buffer Data Field Offset Register (FR_MBDOR36)	16	R/W	See section	54.6.117/ 2579
104A	Message Buffer Data Field Offset Register (FR_MBDOR37)	16	R/W	See section	54.6.117/ 2579
104C	Message Buffer Data Field Offset Register (FR_MBDOR38)	16	R/W	See section	54.6.117/ 2579
104E	Message Buffer Data Field Offset Register (FR_MBDOR39)	16	R/W	See section	54.6.117/ 2579
1050	Message Buffer Data Field Offset Register (FR_MBDOR40)	16	R/W	See section	54.6.117/ 2579
1052	Message Buffer Data Field Offset Register (FR_MBDOR41)	16	R/W	See section	54.6.117/ 2579
1054	Message Buffer Data Field Offset Register (FR_MBDOR42)	16	R/W	See section	54.6.117/ 2579
1056	Message Buffer Data Field Offset Register (FR_MBDOR43)	16	R/W	See section	54.6.117/ 2579
1058	Message Buffer Data Field Offset Register (FR_MBDOR44)	16	R/W	See section	54.6.117/ 2579
105A	Message Buffer Data Field Offset Register (FR_MBDOR45)	16	R/W	See section	54.6.117/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
105C	Message Buffer Data Field Offset Register (FR_MBDOR46)	16	R/W	See section	54.6.117/ 2579
105E	Message Buffer Data Field Offset Register (FR_MBDOR47)	16	R/W	See section	54.6.117/ 2579
1060	Message Buffer Data Field Offset Register (FR_MBDOR48)	16	R/W	See section	54.6.117/ 2579
1062	Message Buffer Data Field Offset Register (FR_MBDOR49)	16	R/W	See section	54.6.117/ 2579
1064	Message Buffer Data Field Offset Register (FR_MBDOR50)	16	R/W	See section	54.6.117/ 2579
1066	Message Buffer Data Field Offset Register (FR_MBDOR51)	16	R/W	See section	54.6.117/ 2579
1068	Message Buffer Data Field Offset Register (FR_MBDOR52)	16	R/W	See section	54.6.117/ 2579
106A	Message Buffer Data Field Offset Register (FR_MBDOR53)	16	R/W	See section	54.6.117/ 2579
106C	Message Buffer Data Field Offset Register (FR_MBDOR54)	16	R/W	See section	54.6.117/ 2579
106E	Message Buffer Data Field Offset Register (FR_MBDOR55)	16	R/W	See section	54.6.117/ 2579
1070	Message Buffer Data Field Offset Register (FR_MBDOR56)	16	R/W	See section	54.6.117/ 2579
1072	Message Buffer Data Field Offset Register (FR_MBDOR57)	16	R/W	See section	54.6.117/ 2579
1074	Message Buffer Data Field Offset Register (FR_MBDOR58)	16	R/W	See section	54.6.117/ 2579
1076	Message Buffer Data Field Offset Register (FR_MBDOR59)	16	R/W	See section	54.6.117/ 2579
1078	Message Buffer Data Field Offset Register (FR_MBDOR60)	16	R/W	See section	54.6.117/ 2579
107A	Message Buffer Data Field Offset Register (FR_MBDOR61)	16	R/W	See section	54.6.117/ 2579
107C	Message Buffer Data Field Offset Register (FR_MBDOR62)	16	R/W	See section	54.6.117/ 2579
107E	Message Buffer Data Field Offset Register (FR_MBDOR63)	16	R/W	See section	54.6.117/ 2579
1080	Message Buffer Data Field Offset Register (FR_MBDOR64)	16	R/W	See section	54.6.117/ 2579
1082	Message Buffer Data Field Offset Register (FR_MBDOR65)	16	R/W	See section	54.6.117/ 2579
1084	Message Buffer Data Field Offset Register (FR_MBDOR66)	16	R/W	See section	54.6.117/ 2579
1086	Message Buffer Data Field Offset Register (FR_MBDOR67)	16	R/W	See section	54.6.117/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1088	Message Buffer Data Field Offset Register (FR_MBDOR68)	16	R/W	See section	54.6.117/ 2579
108A	Message Buffer Data Field Offset Register (FR_MBDOR69)	16	R/W	See section	54.6.117/ 2579
108C	Message Buffer Data Field Offset Register (FR_MBDOR70)	16	R/W	See section	54.6.117/ 2579
108E	Message Buffer Data Field Offset Register (FR_MBDOR71)	16	R/W	See section	54.6.117/ 2579
1090	Message Buffer Data Field Offset Register (FR_MBDOR72)	16	R/W	See section	54.6.117/ 2579
1092	Message Buffer Data Field Offset Register (FR_MBDOR73)	16	R/W	See section	54.6.117/ 2579
1094	Message Buffer Data Field Offset Register (FR_MBDOR74)	16	R/W	See section	54.6.117/ 2579
1096	Message Buffer Data Field Offset Register (FR_MBDOR75)	16	R/W	See section	54.6.117/ 2579
1098	Message Buffer Data Field Offset Register (FR_MBDOR76)	16	R/W	See section	54.6.117/ 2579
109A	Message Buffer Data Field Offset Register (FR_MBDOR77)	16	R/W	See section	54.6.117/ 2579
109C	Message Buffer Data Field Offset Register (FR_MBDOR78)	16	R/W	See section	54.6.117/ 2579
109E	Message Buffer Data Field Offset Register (FR_MBDOR79)	16	R/W	See section	54.6.117/ 2579
10A0	Message Buffer Data Field Offset Register (FR_MBDOR80)	16	R/W	See section	54.6.117/ 2579
10A2	Message Buffer Data Field Offset Register (FR_MBDOR81)	16	R/W	See section	54.6.117/ 2579
10A4	Message Buffer Data Field Offset Register (FR_MBDOR82)	16	R/W	See section	54.6.117/ 2579
10A6	Message Buffer Data Field Offset Register (FR_MBDOR83)	16	R/W	See section	54.6.117/ 2579
10A8	Message Buffer Data Field Offset Register (FR_MBDOR84)	16	R/W	See section	54.6.117/ 2579
10AA	Message Buffer Data Field Offset Register (FR_MBDOR85)	16	R/W	See section	54.6.117/ 2579
10AC	Message Buffer Data Field Offset Register (FR_MBDOR86)	16	R/W	See section	54.6.117/ 2579
10AE	Message Buffer Data Field Offset Register (FR_MBDOR87)	16	R/W	See section	54.6.117/ 2579
10B0	Message Buffer Data Field Offset Register (FR_MBDOR88)	16	R/W	See section	54.6.117/ 2579
10B2	Message Buffer Data Field Offset Register (FR_MBDOR89)	16	R/W	See section	54.6.117/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10B4	Message Buffer Data Field Offset Register (FR_MBDOR90)	16	R/W	See section	54.6.117/ 2579
10B6	Message Buffer Data Field Offset Register (FR_MBDOR91)	16	R/W	See section	54.6.117/ 2579
10B8	Message Buffer Data Field Offset Register (FR_MBDOR92)	16	R/W	See section	54.6.117/ 2579
10BA	Message Buffer Data Field Offset Register (FR_MBDOR93)	16	R/W	See section	54.6.117/ 2579
10BC	Message Buffer Data Field Offset Register (FR_MBDOR94)	16	R/W	See section	54.6.117/ 2579
10BE	Message Buffer Data Field Offset Register (FR_MBDOR95)	16	R/W	See section	54.6.117/ 2579
10C0	Message Buffer Data Field Offset Register (FR_MBDOR96)	16	R/W	See section	54.6.117/ 2579
10C2	Message Buffer Data Field Offset Register (FR_MBDOR97)	16	R/W	See section	54.6.117/ 2579
10C4	Message Buffer Data Field Offset Register (FR_MBDOR98)	16	R/W	See section	54.6.117/ 2579
10C6	Message Buffer Data Field Offset Register (FR_MBDOR99)	16	R/W	See section	54.6.117/ 2579
10C8	Message Buffer Data Field Offset Register (FR_MBDOR100)	16	R/W	See section	54.6.117/ 2579
10CA	Message Buffer Data Field Offset Register (FR_MBDOR101)	16	R/W	See section	54.6.117/ 2579
10CC	Message Buffer Data Field Offset Register (FR_MBDOR102)	16	R/W	See section	54.6.117/ 2579
10CE	Message Buffer Data Field Offset Register (FR_MBDOR103)	16	R/W	See section	54.6.117/ 2579
10D0	Message Buffer Data Field Offset Register (FR_MBDOR104)	16	R/W	See section	54.6.117/ 2579
10D2	Message Buffer Data Field Offset Register (FR_MBDOR105)	16	R/W	See section	54.6.117/ 2579
10D4	Message Buffer Data Field Offset Register (FR_MBDOR106)	16	R/W	See section	54.6.117/ 2579
10D6	Message Buffer Data Field Offset Register (FR_MBDOR107)	16	R/W	See section	54.6.117/ 2579
10D8	Message Buffer Data Field Offset Register (FR_MBDOR108)	16	R/W	See section	54.6.117/ 2579
10DA	Message Buffer Data Field Offset Register (FR_MBDOR109)	16	R/W	See section	54.6.117/ 2579
10DC	Message Buffer Data Field Offset Register (FR_MBDOR110)	16	R/W	See section	54.6.117/ 2579
10DE	Message Buffer Data Field Offset Register (FR_MBDOR111)	16	R/W	See section	54.6.117/ 2579

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10E0	Message Buffer Data Field Offset Register (FR_MBDOR112)	16	R/W	See section	54.6.117/ 2579
10E2	Message Buffer Data Field Offset Register (FR_MBDOR113)	16	R/W	See section	54.6.117/ 2579
10E4	Message Buffer Data Field Offset Register (FR_MBDOR114)	16	R/W	See section	54.6.117/ 2579
10E6	Message Buffer Data Field Offset Register (FR_MBDOR115)	16	R/W	See section	54.6.117/ 2579
10E8	Message Buffer Data Field Offset Register (FR_MBDOR116)	16	R/W	See section	54.6.117/ 2579
10EA	Message Buffer Data Field Offset Register (FR_MBDOR117)	16	R/W	See section	54.6.117/ 2579
10EC	Message Buffer Data Field Offset Register (FR_MBDOR118)	16	R/W	See section	54.6.117/ 2579
10EE	Message Buffer Data Field Offset Register (FR_MBDOR119)	16	R/W	See section	54.6.117/ 2579
10F0	Message Buffer Data Field Offset Register (FR_MBDOR120)	16	R/W	See section	54.6.117/ 2579
10F2	Message Buffer Data Field Offset Register (FR_MBDOR121)	16	R/W	See section	54.6.117/ 2579
10F4	Message Buffer Data Field Offset Register (FR_MBDOR122)	16	R/W	See section	54.6.117/ 2579
10F6	Message Buffer Data Field Offset Register (FR_MBDOR123)	16	R/W	See section	54.6.117/ 2579
10F8	Message Buffer Data Field Offset Register (FR_MBDOR124)	16	R/W	See section	54.6.117/ 2579
10FA	Message Buffer Data Field Offset Register (FR_MBDOR125)	16	R/W	See section	54.6.117/ 2579
10FC	Message Buffer Data Field Offset Register (FR_MBDOR126)	16	R/W	See section	54.6.117/ 2579
10FE	Message Buffer Data Field Offset Register (FR_MBDOR127)	16	R/W	See section	54.6.117/ 2579
1100	Message Buffer Data Field Offset Register (FR_MBDOR128)	16	R/W	See section	54.6.117/ 2579
1102	Message Buffer Data Field Offset Register (FR_MBDOR129)	16	R/W	See section	54.6.117/ 2579
1104	Message Buffer Data Field Offset Register (FR_MBDOR130)	16	R/W	See section	54.6.117/ 2579
1106	Message Buffer Data Field Offset Register (FR_MBDOR131)	16	R/W	See section	54.6.117/ 2579
1108	LRAM ECC Error Test Register (FR_LEETR0)	16	R/W	See section	54.6.118/ 2580
110A	LRAM ECC Error Test Register (FR_LEETR1)	16	R/W	See section	54.6.118/ 2580

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
110C	LRAM ECC Error Test Register (FR_LEETR2)	16	R/W	See section	54.6.118/ 2580
110E	LRAM ECC Error Test Register (FR_LEETR3)	16	R/W	See section	54.6.118/ 2580
1110	LRAM ECC Error Test Register (FR_LEETR4)	16	R/W	See section	54.6.118/ 2580
1112	LRAM ECC Error Test Register (FR_LEETR5)	16	R/W	See section	54.6.118/ 2580

54.6.1 Module Version Register (FR_MVR)

This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHIVER								PEVER							
Write																
Reset	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0

FR_MVR field descriptions

Field	Description
0–7 CHIVER	CHI Version Number. This field provides the version number of the controller host interface.
8–15 PEVER	PE Version Number. This field provides the version number of the protocol engine.

54.6.2 Module Configuration Register (FR_MCR)

This register defines the global configuration of the CC.

Write: SBFF, SCM, CHB, CHA, ECCE, FUM, FAM , CLKSEL , BITRATE: Disabled Mode and SFFE, MEN: Disabled Mode or POC: config

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read Write	MEN	SBFF	SCM	CHB	CHA	SFFE	ECCE	Reserved
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	FUM	FAM	0	CLKSEL	BITRATE			0
Write								
Reset	0	0	0	0	0	0	0	0

FR_MCR field descriptions

Field	Description
0 MEN	Module Enable. This bit indicates whether or not the CC is in the Disabled Mode. The application requests the CC to leave the Disabled Mode by writing 1 to this bit Before leaving the Disabled Mode, the application must configure the SCM, SBFF, CHB, CHA, TMODE, BITRATE values. For details see Modes of Operation. **NOTE:** If the CC is enabled it can only be disabled during mode: POC:default config. 0 Write: only during POC:default config, CC disable Read: CC disabled 1 Write: enable CC Read: CC enabled
1 SBFF	System Bus Failure Freeze This bit controls the behavior of the CC in case of a system bus failure. 0 Continue normal operation 1 Transition to freeze mode
2 SCM	Single Channel Device Mode. This control bit defines the channel device mode of the CC as described in Channel Device Modes. 0 CC works in dual channel device mode 1 CC works in single channel device mode
3 CHB	Channel B Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the following table.

Table 54-7. FlexRay channel selection

SCM	CHB	CHA	Description
Dual Channel Device Modes			
0	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	1	0	reserved
	1	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B
Single Channel Device Mode			

Table continues on the next page...

FR_MCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
Table 54-7. FlexRay channel selection (continued)
<table>
<tr><th>SCM</th><th>CHB</th><th>CHA</th><th>Description</th></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC</td></tr>
<tr><td>0</td><td>1</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC</td></tr>
<tr><td>1</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table>
</td></tr>
<tr><td>4
CHA</td><td>Channel A Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the FlexRay channel selection table as seen above.</td></tr>
<tr><td>5
SFFE</td><td>Synchronization Frame Filter Enable. This bit controls the filtering for received synchronization frames. For details see Sync Frame Filtering .

0 Synchronization frame filtering disabled
1 Synchronization frame filtering enabled</td></tr>
<tr><td>6
ECCE</td><td>ECC Functionality Enable. This bit controls the ECC memory error detection functionality. For details see Memory Content Fault Detection .

0 ECC functionality (injection, detection, reporting, response) disabled
1 ECC functionality enabled</td></tr>
<tr><td>7
Reserved</td><td>Reserved bit, will not be changed. Application must not write any value different from the reset value.

This field is reserved.</td></tr>
<tr><td>8
FUM</td><td>FIFO Update Mode. This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (see FIFO Update)

0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF]
1 FIFOA/FIFOB) is not updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]</td></tr>
<tr><td>9
FAM</td><td>FIFO Address Mode. This bit controls the location of the system memory base address for the FIFOs. (see FIFO Configuration)

0 FIFO Base Address located in System Memory Base Address High Register (FR_SYMBADHR))
1 FIFO Base Address located in Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)</td></tr>
<tr><td>10
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11
CLKSEL</td><td>Protocol Engine Clock Source Select. This bit is used to select the clock source for the protocol engine.

0 PE clock source is generated by on-chip crystal oscillator.
1 PE clock source is generated by on-chip PLL.</td></tr>
</table>

Table continues on the next page...

FR_MCR field descriptions (continued)

Field	Description
12–14 BITRATE	FlexRay Bus Bit Rate. This bit field defines the FlexRay Bus Bit Rate. 000 10.0 Mbit/sec 001 5.0 Mbit/sec 010 2.5 Mbit/sec 011 8.0 Mbit/sec 100 reserved 101 reserved 110 reserved 111 reserved
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.6.3 System Memory Base Address High Register (FR_SYMBADHR)

NOTE

The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADHR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADHR register contains the most significant bits of the base address.

Write: Disabled Mode

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address high. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

54.6.4 System Memory Base Address Low Register (FR_SYMBADLR)

NOTE

The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADLR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADLR register contains the least significant bits of the base address.

Write: Disabled Mode

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SMBA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address low. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.6.5 Strobe Signal Control Register (FR_STBSCR)

Write: Anytime

This register is used to assign the individual protocol timing related strobe signals given in Table 54-8 to the external strobe ports. Each strobe signal can be assigned to at most one strobe port . Each write access to registers overwrites the previously written ENB and STBPSEL values for the signal indicated by SEL . If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR and presented at the strobe port . If no strobe signal is assigned to a strobe port, the strobe port carries logic 0 . For more detailed and timing information refer to Strobe Signal Support .

NOTE

In single channel device mode, channel B related strobe signals are undefined and should not be assigned to the strobe ports.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			SEL				0			ENB	0		STBPSEL	
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_STBSCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SEL	Strobe Signal Select. This control field selects one of the strobe signals given in Table 54-8 to be enabled or disabled and assigned to one of the four strobe ports given in the following table.

Table 54-8. Strobe Signal Mapping

SEL		Description	Channel	Type	Offset (Given in PE clock cycles)	Reference
dec	hex					
0	0x0	arm	-	value	+1	MT start
1	0x1	mt	-	value	+1	MT start
2	0x2	cycle start	-	pulse	0	MT start
3	0x3	minislot start	-	pulse	0	MT start
4	0x4	slot start	A	pulse	0	MT start
5	0x5		B			
6	0x6	receive data after glitch filtering	A	value	+4	FR_A_RX
7	0x7		B			FR_B_RX
8	0x8	channel idle indicator	A	level	+5	FR_A_RX
9	0x9		B			FR_B_RX
10	0xA	syntax error detected	A	pulse	+4	FR_A_RX
11	0xB		B			FR_B_RX
12	0xC	content error detected	A	level	+4	FR_A_RX
13	0xD		B			FR_B_RX
14	0xE	receive FIFO almost-full interrupt signals	A	value	n.a.	RX FIFO A Almost Full Interrupt

Table continues on the next page...

FR_STBSCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
Table 54-8. Strobe Signal Mapping (continued)
<table>
<tr><th colspan="2">SEL</th><th>Description</th><th>Channel</th><th>Type</th><th>Offset (Given in PE clock cycles)</th><th>Reference</th></tr>
<tr><th>dec</th><th>hex</th><th></th><th></th><th></th><th></th><th></th></tr>
<tr><td>15</td><td>0xF</td><td></td><td>B</td><td></td><td></td><td>RX FIFO B Almost Full Interrupt</td></tr>
<tr><td colspan="7">Write: Disabled Mode</td></tr>
</table>
</td></tr>
<tr><td>8–10
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11
ENB</td><td>Strobe Signal Enable. The control bit is used to enable and to disable the strobe signal selected by STBSSEL.

0 Strobe signal is disabled and not assigned to any strobe port.
1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.</td></tr>
<tr><td>12–13
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>14–15
STBPSEL</td><td>Strobe Port Select. This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation.

00 assign selected signal to FR_DBG[0]
01 assign selected signal to FR_DBG[1]
10 assign selected signal to FR_DBG[2]
11 assign selected signal to FR_DBG[3]</td></tr>
</table>

54.6.6 Message Buffer Data Size Register (FR_MBDSR)

Write: POC:config

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	MBSEG2DS							0	MBSEG1DS						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBDSR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 MBSEG2DS	Message Buffer Segment 2 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the second message buffer segment.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBSEG1DS	Message Buffer Segment 1 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the first message buffer segment.

54.6.7 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Write: POC:config

This register is used to define the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Address: 0h base + Eh offset = Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	LAST_MB_SEG1							0	LAST_MB_UTIL						
Write																
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

FR_MBSSUTR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 LAST_MB_SEG1	Last Message Buffer In Segment 1. This field defines the message buffer number of the last individual message buffer that is assigned to the first message buffer segment. The individual message buffers in the first segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with n <= LAST_MB_SEG1. The first message buffer segment contains LAST_MB_SEG1+1 individual message buffers. **NOTE:** The first message buffer segment contains at least one individual message buffer. The individual message buffers in the second message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with LAST_MB_SEG1 < n < 128 .

Table continues on the next page...

FR_MBSSUTR field descriptions (continued)

Field	Description
	NOTE: If LAST_MB_SEG1 = 127 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LAST_MB_UTIL	Last Message Buffer Utilized. This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number n <= LAST_MB_UTIL. **NOTE:** If LAST_MB_UTIL=LAST_MB_SEG1 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

54.6.8 PE DRAM Access Register (FR_PEDRAR)

The PEDRAR register is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this registers initiates a read or write operation on the PE DRAM. The access done status bit DAD is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRDR are written into the PE DRAM, read back from the PE DRAM and are stored into the FR_PEDRDR.

In case of an PE DRAM read access, the requested data are read from PE DRAM and stored into the FR_PEDRDR.

For a detailed description refer to Memory Content Fault Detection

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	INST				ADDR											DAD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRAR field descriptions

Field	Description
0–3 INST	PE DRAM Access Instruction. This field defines the operation to be executed on the PE DRAM. 0011 PE DRAM write: Write FR_PEDRDR[DATA] to PE DRAM address ADDR (16 bit) 0101 PE DRAM read: Read Data from PE DRAM address ADDR (16 bit) into FR_PEDRDR[DATA] other reserved

Table continues on the next page...

FR_PEDRAR field descriptions (continued)

Field	Description
4–14 ADDR	PE DRAM Access Address. This field defines the address in the PE DRAM to be written to or read from.
15 DAD	PE DRAM Access Done. This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done

54.6.9 PE DRAM Data Register (FR_PEDRDR)

This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

Address: 0h base + 12h offset = 12h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRDR field descriptions

Field	Description
0–15 DATA	Data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

54.6.10 Protocol Operation Control Register (FR_POCR)

The application uses this register to issue

- protocol control commands
- external clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, see Protocol Control Command Execution .

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to External Clock Synchronization .

Write: Normal Mode

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			EOC_AP		ERC_AP		BSY_WMC	0			POCCMD			
Write	WME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_POCR field descriptions

Field	Description
0 WME	Write Mode External Correction. This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write. 1 No write to EOC_AP and ERC_AP fields on register write.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 EOC_AP	External Offset Correction Application. This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29). 00 do not apply external offset correction value 01 reserved 10 subtract external offset correction value 11 add external offset correction value
6–7 ERC_AP	External Rate Correction Application. This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21) 00 do not apply external rate correction value 01 reserved 10 subtract external rate correction value 11 add external rate correction value
8 BSY_WMC	**NOTE:** The name and function of this field varies depending on whether it is being read or written. BSY (Read-Only). Protocol Control Command Busy. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. WMC (Write-Only) Protocol Control Command Write. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. 0 (Write-Only) Write to POCCMD field on register write. 1 (Write-Only) Do not write to POCCMD field on register write. 0 (Read-Only) Command write idle, command accepted and ready to receive new protocol command. 1 (Read-Only) Command write busy, command not yet accepted, not ready to receive new protocol command.

Table continues on the next page...

FR_POCR field descriptions (continued)

Field	Description
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 POCCMD	Protocol Control Command. The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTART. Immediately activate capability of node to cold start cluster. 0001 ALL_SLOTS. Delayed transition to the all slots transmission mode. (Delayed means on completion of current communication cycle.) 0010 CONFIG. Immediately transition to the POC:config state. 0011 FREEZE. Immediately transition to the POC:halt state. 0100 READY, CONFIG_COMPLETE. Immediately transition to the POC:ready state. 0101 RUN. Immediately transition to the POC:startup start state. 0110 DEFAULT_CONFIG. Immediately transition to the POC:default config state. 0111 HALT. Delayed transition to the POC:halt state 1000 WAKEUP. Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

54.6.11 Global Interrupt Flag and Enable Register (FR_GIFER)

Write: Normal Mode

This register provides the means to control some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables . The generation scheme for these flags is depicted in Figure 54-33 . For more details on interrupt generation, see Interrupt Support . These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Address: 0h base + 16h offset = 16h

Bit	0	1	2	3	4	5	6	7
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write				w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MIE	PRIE	CHIE	WUPIE	FAFBIE	FAFAIE	RBIE	TBIE
Reset	0	0	0	0	0	0	0	0

FR_GIFER field descriptions

Field	Description
0 MIF	Module Interrupt Flag. This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set.
1 PRIF	Protocol Interrupt Flag. This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set. 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set.
2 CHIF	CHI Interrupt Flag. This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the chi error interrupt enable bit FR_GIFER[CHIE] are set . 0 All CHI error flags are equal to 0 or the chi error interrupt is disabled. 1 At least one CHI error flag and the chi error interrupt enable are is set.
3 WUPIF	Wakeup Interrupt Flag. This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3). 0 No Wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus
4 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO B, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO B almost full event has occurred
5 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO A, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO A almost full event has occurred

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
6 RBIF	Receive Message Buffer Interrupt Flag. This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application can not clear this interrupt flag directly, instead it is cleared by the CC when all of the interrupt flags MBIF of the individual receive message buffers are cleared by the application or if the application has cleared the related interrupt enables bit MBIE. 0 None of the individual receive message buffers has the MBIF and MBIE flag set. 1 At least one individual receive message buffer has the MBIF and MBIE flag set.
7 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application can not clear this interrupt flag directly, instead, this interrupt flag is cleared by the CC when either all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application or the application has cleared the related interrupt enables bit MBIE. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set. 1 At least one individual transmit message buffer has the MBIF and MBIE flag set.
8 MIE	Module Interrupt Enable. This bit controls if the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
9 PRIE	Protocol Interrupt Enable. This bit controls if the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
10 CHIE	CHI Interrupt Enable. This bit controls if the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
11 WUPIE	Wakeup Interrupt Enable. This bit controls if the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
12 FAFBIE	Receive FIFO Channel B Almost Full Interrupt Enable. This bit controls if the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 FAFAIE	Receive FIFO Channel A Almost Full Interrupt Enable. This bit controls if the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 RBIE	Receive Message Buffer Interrupt Enable. This bit controls if the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
15 TBIE	Transmit Message Buffer Interrupt Enable. This bit controls if the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

54.6.12 Protocol Interrupt Flag Register 0 (FR_PIFR0)

Write: Normal Mode

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	FATL_IF	INTL_IF	ILCF_IF	CSA_IF	MRC_IF	MOC_IF	CCL_IF	MXS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MTX_IF	LTXB_IF	LTXA_IF	TBVB_IF	TBVA_IF	TI2_IF	TI1_IF	CYS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PIFR0 field descriptions

Field	Description
0 FATL_IF	Fatal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. The fatal protocol errors are: 1) pLatestTx violation, as described in the MAC process of the FlexRay protocol 2) transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event. 1 Fatal protocol error detected.
1 INTL_IF	Internal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event. 1 Internal protocol error detected.
2 ILCF_IF	Illegal Protocol Configuration Interrupt Flag. This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the POC:halt state immediately.

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
	The protocol engine checks the listen_timeout value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of listen_timeout is equal to zero, the protocol configuration setting is considered as illegal. 0 No such event. 1 Illegal protocol configuration detected.
3 CSA_IF	Cold Start Abort Interrupt Flag. This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the coldstart_attempts field in the Protocol Configuration Register 3 (FR_PCR3). 0 No such event. 1 Cold start aborted and no more coldstart attempts allowed.
4 MRC_IF	Missing Rate Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
5 MOC_IF	Missing Offset Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event. 1 Insufficient number of measurements for offset correction detected.
6 CCL_IF	Clock Correction Limit Reached Interrupt Flag. This flag is set when the internal calculated offset or rate calculation values have reached or exceeded its configured thresholds as given by the offset_coorection_out field in the Protocol Configuration Register 9 (FR_PCR9) and the rate_correction_out field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event. 1 Offset or rate correction limit reached.
7 MXS_IF	Max Sync Frames Detected Interrupt Flag. This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the node_sync_max field in the Protocol Configuration Register 30 (FR_PCR30). **NOTE:** Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account. 0 No such event. 1 More than node_sync_max sync frames detected.
8 MTX_IF	Media Access Test Symbol Received Interrupt Flag. This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event. 1 MTS symbol received.
9 LTXB_IF	pLatestTx Violation on Channel B Interrupt Flag. This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation, as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel B.

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
10 LTXA_IF	pLatestTx Violation on Channel A Interrupt Flag. This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel A.
11 TBVB_IF	Transmission across boundary on channel B Interrupt Flag. This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel B.
12 TBVA_IF	Transmission across boundary on channel A Interrupt Flag. This flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel A.
13 TI2_IF	Timer 2 Expired Interrupt Flag. This flag is set whenever timer 2 expires. 0 No such event. 1 Timer 2 has reached its time limit.
14 TI1_IF	Timer 1 Expired Interrupt Flag. This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
15 CYS_IF	Cycle Start Interrupt Flag. This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started.

54.6.13 Protocol Interrupt Flag Register 1 (FR_PIFR1)

Write: Normal Mode

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	EMC_IF	IPC_IF	PECF_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IF	ODT_IF	0			
Write			w1c	w1c				
Reset	0	0	0	0	0	0	0	0

FR_PIFR1 field descriptions

Field	Description
0 EMC_IF	Error Mode Changed Interrupt Flag. This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC. 0 No such event. 1 ERRMODE field changed.
1 IPC_IF	Illegal Protocol Control Command Interrupt Flag. This flag is set when the PE tries to execute a protocol control command, which was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR), and detects that this protocol control command is not allowed in the current protocol state. In this case the command is not executed. For more details, see Protocol Control Command Execution . 0 No such event. 1 Illegal protocol control command detected.
2 PECF_IF	Protocol Engine Communication Failure Interrupt Flag. This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event. 1 Protocol Engine Communication Failure detected.
3 PSC_IF	Protocol State Changed Interrupt Flag. This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event. 1 Protocol state changed.
4 SSI3_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
5 SSI2_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
6 SSI1_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
7 SSI0_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_PIFR1 field descriptions (continued)

Field	Description
10 EVT_IF	Even Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the even cycle. 0 No such event. 1 Sync frame measurement table written
11 ODT_IF	Odd Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the odd cycle. 0 No such event. 1 Sync frame measurement table written
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.6.14 Protocol Interrupt Enable Register 0 (FR_PIER0)

Write: Anytime

This register defines whether or not the individual interrupt flags defined in the Protocol Interrupt Flag Register 0 (FR_PIFR0) can generate a protocol interrupt request.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read Write	FATL_IE	INTL_IE	ILCF_IE	CSA_IE	MRC_IE	MOC_IE	CCL_IE	MXS_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MTX_IE	LTXB_IE	LTXA_IE	TBVB_IE	TBVA_IE	TI2_IE	TI1_IE	CYS_IE
Reset	0	0	0	0	0	0	0	0

FR_PIER0 field descriptions

Field	Description
0 FATL_IE	Fatal Protocol Error Interrupt Enable. This bit controls FATL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 INTL_IE	Internal Protocol Error Interrupt Enable. This bit controls INTL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 ILCF_IE	Illegal Protocol Configuration Interrupt Enable. This bit controls ILCF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER0 field descriptions (continued)

Field	Description
3 CSA_IE	Cold Start Abort Interrupt Enable. This bit controls CSA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 MRC_IE	Missing Rate Correction Interrupt Enable. This bit controls MRC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 MOC_IE	Missing Offset Correction Interrupt Enable. This bit controls MOC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 CCL_IE	Clock Correction Limit Reached Interrupt Enable. This bit controls CCL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 MXS_IE	Max Sync Frames Detected Interrupt Enable. This bit controls MXS_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8 MTX_IE	Media Access Test Symbol Received Interrupt Enable. This bit controls MTX_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
9 LTXB_IE	pLatestTx Violation on Channel B Interrupt Enable. This bit controls LTXB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
10 LTXA_IE	pLatestTx Violation on Channel A Interrupt Enable. This bit controls LTXA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 TBVB_IE	Transmission across boundary on channel B Interrupt Enable. This bit controls TBVB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12 TBVA_IE	Transmission across boundary on channel A Interrupt Enable. This bit controls TBVA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
13 TI2_IE	Timer 2 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
14 TI1_IE	Timer 1 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation.

Table continues on the next page...

FR_PIER0 field descriptions (continued)

Field	Description
	0 interrupt request generation disabled 1 interrupt request generation enabled
15 CYS_IE	Cycle Start Interrupt Enable. This bit controls CYC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

54.6.15 Protocol Interrupt Enable Register 1 (FR_PIER1)

Write: Anytime

This register defines whether or not the individual interrupt flags defined in Protocol Interrupt Flag Register 1 (FR_PIFR1) can generate a protocol interrupt request.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read Write	EMC_IE	IPC_IE	PECF_IE	PSC_IE	SSI3_IE	SSI2_IE	SSI1_IE	SSI0_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IE	ODT_IE	0			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PIER1 field descriptions

Field	Description
0 EMC_IE	Error Mode Changed Interrupt Enable. This bit controls EMC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 IPC_IE	Illegal Protocol Control Command Interrupt Enable. This bit controls IPC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 PECF_IE	Protocol Engine Communication Failure Interrupt Enable. This bit controls PECF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 PSC_IE	Protocol State Changed Interrupt Enable. This bit controls PSC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER1 field descriptions (continued)

Field	Description
4 SSI3_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 SSI2_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 SSI1_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 SSI0_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IE	Even Cycle Table Written Interrupt Enable. This bit controls EVT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 ODT_IE	Odd Cycle Table Written Interrupt Enable. This bit controls ODT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.6.16 CHI Error Flag Register (FR_CHIERFR)

Write: Normal Mode

This register holds the CHI related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit CHIE in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7
Read	FRLB_EF	FRLA_EF	PCMI_EF	FOVB_EF	FOVA_EF	MBS_EF	MBU_EF	LCK_EF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0	SBCF_EF	FID_EF	DPL_EF	SPL_EF	NML_EF	NMF_EF	ILSA_EF
Write		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_CHIERFR field descriptions

Field	Description
0 FRLB_EF	Frame Lost Channel B Error Flag. This flag is set if a complete frame was received on channel B but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such event 1 Frame lost on channel B detected
1 FRLA_EF	Frame Lost Channel A Error Flag. This flag is set if a complete frame was received on channel A but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected
2 PCMI_EF	Protocol Command Ignored Error Flag. This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCR) while the BSY flag is equal to 1. In this case the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
3 FOVB_EF	Receive FIFO Overrun Channel B Error Flag. This flag is set when an overrun of the FIFO for channel B occurred. This error occurs if a semantically valid frame was received on channel B and matches the all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
4 FOVA_EF	Receive FIFO Overrun Channel A Error Flag. This flag is set when an overrun of the FIFO for channel A occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
5 MBS_EF	Message Buffer Search Error Flag. This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 131 is detected in the FR_MBIDXR[MBIDX] field of an found message buffer or in one of the FR_RSBIR[RSBIDX] fields. Refer to Message Buffer Search Error for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
6 MBU_EF	Message Buffer Utilization Error Flag. This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled
7 LCK_EF	Lock Error Flag. This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again. 0 No such error 1 Lock error detected
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SBCF_EF	System Bus Communication Failure Error Flag. This flag is set if a system bus access was not finished within the required amount of time (see System Bus Access Timeout). 0 No such event 1 System bus access not finished in time
10 FID_EF	Frame ID Error Flag. This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register . 0 No such error occurred 1 Frame ID error occurred
11 DPL_EF	Dynamic Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding protocol configuration register field max_payload_length_dynamic in the Protocol Configuration Register 24 (FR_PCR24). 0 No such error occurred 1 Dynamic payload length error occurred
12 SPL_EF	Static Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding protocol configuration register field payload_length_static in the Protocol Configuration Register 19 (FR_PCR19). 0 No such error occurred 1 Static payload length error occurred
13 NML_EF	Network Management Length Error Flag. This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR). In this case the received part of the Network Management Vector will be used to update the Network Management Vector. 0 No such error occurred 1 Network management length error occurred
14 NMF_EF	Network Management Frame Error Flag. This flag is set if a received message in the static segment with a Preamble Indicator flag PP asserted has its Null Frame indicator flag NF asserted as well. In this case, the

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
	Global Network Management Registers (see Network Management Vector Registers (FR_NMVR0-FR_NMVR5)) are not updated. 0 No such error occurred 1 Network management frame error occurred
15 ILSA_EF	Illegal System Bus Address Error Flag. This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (see System Bus Illegal Address Access). 0 No such event 1 Illegal system bus address accessed

54.6.17 Message Buffer Interrupt Vector Register (FR_MBIVEC)

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their interrupt status flag MBIF and interrupt enable MBIE bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Address: 0h base + 22h offset = 22h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	TBIVEC							0	RBIVEC						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBIVEC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 TBIVEC	Transmit Buffer Interrupt Vector. This field provides the number of the lowest numbered enabled transmit message buffer that has its interrupt status flag MBIF and its interrupt enable bit MBIE set. If there is no transmit message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 RBIVEC	Receive Buffer Interrupt Vector. This field provides the message buffer number of the lowest numbered receive message buffer which has its interrupt flag MBIF and its interrupt enable bit MBIE asserted. If there is no receive message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.

54.6.18 Channel A Status Error Counter Register (FR_CASERCR)

This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see Slot Status Monitoring.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHAERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CASERCR field descriptions

Field	Description
0–15 CHAERSCNT	Channel A Status Error Counter. This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

54.6.19 Channel B Status Error Counter Register (FR_CBSERCR)

This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring see Slot Status Monitoring.

Address: 0h base + 26h offset = 26h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHBERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CBSERCR field descriptions

Field	Description
0–15 CHBERSCNT	Channel B Status Error Counter. This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

54.6.20 Protocol Status Register 0 (FR_PSR0)

This register provides information about the current protocol status.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ERRMODE		SLOTMODE		0	PROTSTATE			STARTUPSTATE				0	WAKEUPSTATUS		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PSR0 field descriptions

Field	Description
0–1 ERRMODE	Error Mode. protocol related variable: vPOC!ErrorMode. This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE 10 COMM_HALT 11 reserved
2–3 SLOTMODE	Slot Mode. protocol related variable: vPOC!SlotMode. This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 reserved
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 PROTSTATE	Protocol State. protocol related variable: vPOC!State. This field indicates the state of the protocol. 000 POC:default config 001 POC:config 010 POC:wakeup 011 POC:ready 100 POC:normal passive 101 POC:normal active 110 POC:halt 111 POC:startup

Table continues on the next page...

FR_PSR0 field descriptions (continued)

Field	Description
8–11 STARTUPSTATE	Startup State. protocol related variable: vPOC!StartupState. This field indicates the current sub-state of the startup procedure. 0000 reserved 0001 reserved 0010 POC:coldstart collision resolution 0011 POC:coldstart listen 0100 POC:integration consistency check 0101 POC:integration listen 0110 reserved 0111 POC:initialize schedule 1000 reserved 1001 reserved 1010 POC:coldstart consistency check 1011 reserved 1100 reserved 1101 POC:integration coldstart check 1110 POC:coldstart gap 1111 POC:coldstart join
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WAKEUPSTATUS	Wakeup Status. protocol related variable: vPOC!WakeupStatus. This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_ HEADER 100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 reserved

54.6.21 Protocol Status Register 1 (FR_PSR1)

Write: Normal Mode

Address: 0h base + 2Ah offset = 2Ah

Bit	0	1	2	3	4	5	6	7
Read	CSAA	CSP	0	REMCSAT				
Write	w1c							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	CPN	HHR	FRZ	APTAC				
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR1 field descriptions

Field	Description
0 CSAA	Cold Start Attempt Aborted Flag. protocol related event: 'set coldstart abort indicator in CHI' This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
1 CSP	Leading Cold Start Path. This status bit is set when the CC has reached the POC:normal active state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 POC:normal active reached from POC:startup state via leading cold start path
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 REMCSAT	Remaining Coldstart Attempts. protocol related variable: vRemainingColdstartAttempts This field provides the number of remaining cold start attempts that the CC will execute.
8 CPN	Leading Cold Start Path Noise. protocol related variable: vPOC!ColdstartNoise This status bit is set if the CC has reached the POC:normal active state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 POC:normal active state was reached from POC:startup state via noisy leading cold start path
9 HHR	Host Halt Request Pending. protocol related variable: vPOC!CHIHaltRequest This status bit is set when CC receives the HALT command from the application via the Protocol Operation Control Register (FR_POCR). The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 HALT command received

Table continues on the next page...

FR_PSR1 field descriptions (continued)

Field	Description
10 FRZ	Freeze Occurred. protocol related variable: vPOC!Freeze This status bit is set when the CC has reached the POC:halt state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
11–15 APTAC	Allow Passive to Active Counter. protocol related variable: vPOC!vAllowPassivetoActive This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the POC:normal passive state due to an application configured delay to enter POC:normal active state. This delay is defined by the allow_passive_to_active field in the Protocol Configuration Register 12 (FR_PCR12).

54.6.22 Protocol Status Register 2 (FR_PSR2)

This register provides a snapshot of status information about the Network Idle Time NIT, the Symbol Window and the clock synchronization. The NIT related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The Symbol Window related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol window related status bits remain in their reset state. The clock synchronization related CLKCORRFAILCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7
Read	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	STCA	SBVA	SSEA	MTA	CKCORFCNT			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR2 field descriptions

Field	Description
0 NBVB	NIT Boundary Violation on Channel B. protocol related variable: vSS!BViolation for NIT on channel B This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
1 NSEB	NIT Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for NIT on channel B This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
2 STCB	Symbol Window Transmit Conflict on Channel B. protocol related variable: vSS!TxConflict for symbol window on channel B This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
3 SBVB	Symbol Window Boundary Violation on Channel B. protocol related variable: vSS!BViolation for symbol window on channel B This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
4 SSEB	Symbol Window Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for symbol window on channel B This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected
5 MTB	Media Access Test Symbol MTS Received on Channel B. protocol related variable: vSS!ValidMTS for Symbol Window on channel B This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
6 NBVA	NIT Boundary Violation on Channel A. protocol related variable: vSS!BViolation for NIT on channel A This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
7 NSEA	NIT Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for NIT on channel A This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected

Table continues on the next page...

FR_PSR2 field descriptions (continued)

Field	Description
8 STCA	Symbol Window Transmit Conflict on Channel A. protocol related variable: vSS!TxConflict for symbol window on channel A This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
9 SBVA	Symbol Window Boundary Violation on Channel A. protocol related variable: vSS!BViolation for symbol window on channel A This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
10 SSEA	Symbol Window Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for symbol window on channel A This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected
11 MTA	Media Access Test Symbol MTS Received on Channel A. protocol related variable: vSS!ValidMTS for symbol window on channel A This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel A. 1 MTS symbol received 0 No such event
12–15 CKCORFCNT	Clock Correction Failed Counter. protocol related variable: vClockCorrectionFailed This field provides the number of consecutive even/odd communication cycle pairs that have passed without clock synchronization having performed an offset or a rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either max_without_clock_correction_fatal or max_without_clock_correction_passive as defined in the Protocol Configuration Register 8 (FR_PCR8). The CC resets this counter on a hard reset condition, when the protocol enters the POC:normal active state, or when both the rate and offset correction terms have been calculated successfully.

54.6.23 Protocol Status Register 3 (FR_PSR3)

Write: Normal Mode

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Address: 0h base + 2Eh offset = 2Eh

Bit	0	1	2	3	4	5	6	7
Read	0		WUB	ABVB	AACB	ACEB	ASEB	AVFB
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		WUA	ABVA	AACA	ACEA	ASEA	AVFA
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PSR3 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 WUB	Wakeup Symbol Received on Channel B. This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
3 ABVB	Aggregated Boundary Violation on Channel B. This flag is set when a boundary violation has been detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
4 AACB	Aggregated Additional Communication on Channel B. This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
5 ACEB	Aggregated Content Error on Channel B. This flag is set when a content error has been detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
6 ASEB	Aggregated Syntax Error on Channel B. This flag is set when a syntax error has been detected on channel B. Syntax errors are detected in the communication slots, the symbol window and the NIT. 0 No syntax error detected 1 Syntax errors detected
7 AVFB	Aggregated Valid Frame on Channel B. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 1 At least one syntactically valid frame received 0 No syntactically valid frames received
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_PSR3 field descriptions (continued)

Field	Description
10 WUA	Wakeup Symbol Received on Channel A. This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received
11 ABVA	Aggregated Boundary Violation on Channel A. This flag is set when a boundary violation has been detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
12 AACA	Aggregated Additional Communication on Channel A. This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
13 ACEA	Aggregated Content Error on Channel A. This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
14 ASEA	Aggregated Syntax Error on Channel A. This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
15 AVFA	Aggregated Valid Frame on Channel A. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

54.6.24 Macrotick Counter Register (FR_MTCTR)

This register provides the macrotick count of the current communication cycle.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		MTCT													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTCTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 MTCT	Macrotick Counter. protocol related variable: vMacrotick This field provides the macrotick count of the current communication cycle.

54.6.25 Cycle Counter Register (FR_CYCTR)

This register provides the number of the current communication cycle.

Address: 0h base + 32h offset = 32h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0										CYCCNT					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CYCTR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNT	Cycle Counter. protocol related variable: vCycleCounter This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from zero again.

54.6.26 Slot Counter Channel A Register (FR_SLTCTAR)

This register provides the number of the current slot in the current communication cycle for channel A.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTA	Slot Counter Value for Channel A. protocol related variable: vSlotCounter for channel A This field provides the number of the current slot in the current communication cycle.

54.6.27 Slot Counter Channel B Register (FR_SLTCTBR)

This register provides the number of the current slot in the current communication cycle for channel B.

Address: 0h base + 36h offset = 36h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTB	Slot Counter Value for Channel B. protocol related variable: vSlotCounter for channel B This field provides the number of the current slot in the current communication cycle.

54.6.28 Rate Correction Value Register (FR_RTCORVR)

This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RATECORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RTCORVR field descriptions

Field	Description
0–15 RATECORR	Rate Correction Value. protocol related variable: vRateCorrection (before value limitation and external rate correction) This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by rate_correction_out in the Protocol Configuration Register 13 (FR_PCR13), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.

54.6.29 Offset Correction Value Register (FR_OFCORVR)

This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Address: 0h base + 3Ah offset = 3Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	OFFSETCORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_OFCORVR field descriptions

Field	Description
0–15 OFFSETCORR	Offset Correction Value. protocol related variable: vOffsetCorrection (before value limitation and external offset correction) This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by offset_correction_out field in the Protocol Configuration Register 29 (FR_PCR29), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.

54.6.30 Combined Interrupt Flag Register (FR_CIFR)

This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in Figure 54-35 . The individual interrupt flags WUPIF, FAFBIF, and FAFAIF are copies of corresponding flags in the Global Interrupt Flag and Enable Register (FR_GIFER) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the Global Interrupt Flag and Enable Register (FR_GIFER).

NOTE

The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write								
Reset	0	0	0	0	0	0	0	0

FR_CIFR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MIF	Module Interrupt Flag. This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
9 PRIF	Protocol Interrupt Flag. This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
10 CHIF	CHI Interrupt Flag. This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1.

Table continues on the next page...

FR_CIFR field descriptions (continued)

Field	Description
	0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
11 WUPIF	Wakeup Interrupt Flag. Provides the same value as FR_GIFER[WUPIF]
12 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFBIF]
13 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFAIF]
14 RBIF	Receive Message Buffer Interrupt Flag. This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted. 1 At least one individual receive message buffers has the MBIF flag asserted.
15 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted. 1 At least one individual transmit message buffers has the MBIF flag asserted.

54.6.31 System Memory Access Time-Out Register (FR_SYMATOR)

Write: Disabled Mode

Address: 0h base + 3Eh offset = 3Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								TIMEOUT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

FR_SYMATOR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 TIMEOUT	System Memory Access Time-Out. This value defines when a system bus access timout is detected. For a detailed description see Configure System Memory Access Time-Out Register (FR_SYMATOR) and System Bus Access Timeout .

54.6.32 Sync Frame Counter Register (FR_SFCNTR)

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 MT after offset correction start.

NOTE

If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.

NOTE

If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the values SFODB and SFODA.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFEVB				SFEVA				SFODB				SFODA			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFCNTR field descriptions

Field	Description
0–3 SFEVB	Sync Frames Channel B, even cycle. protocol related variable: size of (vsSyncIdListB for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
4–7 SFEVA	Sync Frames Channel A, even cycle. protocol related variable: size of (vsSyncIdListA for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
8–11 SFODB	Sync Frames Channel B, odd cycle. protocol related variable: size of (vsSyncIdListB for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .
12–15 SFODA	Sync Frames Channel A, odd cycle. protocol related variable: size of (vsSyncIdListA for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .

54.6.33 Sync Frame Table Offset Register (FR_SFTOR)

Write: POC:config

This register defines the FlexRay memory area related offset for sync frame tables. For more details, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 42h offset = 42h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFT_OFFSET															0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFTOR field descriptions

Field	Description
0–14 SFT_OFFSET	Sync Frame Table Offset. The offset of the Sync Frame Tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus STF_OFFSET[0] is always 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.6.34 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Write: Normal Mode

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7
Read	0	0	CYCNUM					
Write	ELKT	OLKT						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	ELKS	OLKS	EVAL	OVAL	0	0	SDVEN	SIDEN
Write						OPT		
Reset	0	0	0	0	0	0	0	0

FR_SFTCCSR field descriptions

Field	Description
0 ELKT	Even Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables.
1 OLKT	Odd Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables.

Table continues on the next page...

FR_SFTCCSR field descriptions (continued)

Field	Description
2–7 CYCNUM	Cycle Number. This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.
8 ELKS	Even Cycle Tables Lock Status. This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables. 1 Application has locked the even cycle tables.
9 OLKS	Odd Cycle Tables Lock Status. This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables. 1 Application has locked the odd cycle tables.
10 EVAL	Even Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
11 OVAL	Odd Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 OPT	One Pair Trigger. This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.
14 SDVEN	Sync Frame Deviation Table Enable. This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. **NOTE:** If SDVEN is set to 1, then SIDEN must also be set to 1. 0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area
15 SIDEN	Sync Frame ID Table Enable. This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

54.6.35 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Write: Normal Mode

This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Address: 0h base + 46h offset = 46h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								SYNFRID							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDRFR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SYNFRID	Sync Frame Rejection ID. This field defines the frame ID of a frame that must not be used for clock synchronization. For details see Sync Frame Rejection Filtering .

54.6.36 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

Write: POC:config

This register defines the sync frame acceptance filter value. For details on filtering, see Sync Frame Filtering .

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								FVAL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFVR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FVAL	Filter Value. This field defines the value for the sync frame acceptance filtering.

54.6.37 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

Write: POC:config

This register defines the sync frame acceptance filter mask. For details on filtering see Sync Frame Acceptance Filtering .

Address: 0h base + 4Ah offset = 4Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						FMSK									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFMR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FMSK	Filter Mask. This field defines the mask for the sync frame acceptance filtering.

54.6.38 Network Management Vector Register (FR_NMVR*n*)

Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the Network Management Vector Length Register (FR_NMVLR). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the Network Management Vector Registers (FR_NMVR0-FR_NMVR5), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.

Address: 0h base + 4Ch offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	NMVP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVR*n* field descriptions

Field	Description
0–15 NMVP	Network Management Vector Part. The mapping between the Network Management Vector Registers (FR_NMVR0-FR_NMVR5) and the receive message buffer payload bytes in NMV[0:11] is depicted in the following table.

Table 54-9. Mapping of NMVRn to the Received Payload Bytes NMVn

NMVRn Register	NMVn Received Payload
FR_NMVR0[NMVP[15:8]]	NMV0
FR_NMVR0[NMVP[7:0]]	NMV1
FR_NMVR1[NMVP[15:8]]	NMV2
FR_NMVR1[NMVP[7:0]]	NMV3
...	
FR_NMVR5[NMVP[15:8]]	NMV10
FR_NMVR5[NMVP[7:0]]	NMV11

54.6.39 Network Management Vector Length Register (FR_NMVLR)

Write: POC:config

This register defines the length of the network management vector in bytes.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												NMVL			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVLR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 NMVL	Network Management Vector Length. protocol related variable: gNetworkManagementVectorLength This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

54.6.40 Timer Configuration and Control Register (FR_TICCR)

Write: T2_CFG: POC:config T2_REP, T1_REP, T1SP, T2SP, T1TR, T2TR: Normal Mode

This register is used to configure and control the two timers T1 and T2. For timer details, see Timer Support . The Timer T1 is an absolute timer. The Timer T2 can be configured as an absolute or relative timer.

NOTE

Both timers are deactivated immediately when the protocol enters a state different from POC:normal active or POC:normal passive.

Address: 0h base + 5Ah offset = 5Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2_ CFG	T2_ REP	0	0	0	T2ST	0			T1_ REP	0	0	0	T1ST
Write						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TICCR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 T2_CFG	Timer T2 Configuration. This bit configures the timebase mode of Timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
3 T2_REP	Timer T2 Repetitive Mode. This bit configures the repetition mode of Timer T2. 0 T2 is non repetitive 1 T2 is repetitive
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 T2SP	Timer T2 Stop. This trigger bit is used to stop timer T2. 0 no effect 1 stop timer T2
6 T2TR	Timer T2 Trigger. This trigger bit is used to start timer T2. 0 no effect 1 start timer T2
7 T2ST	Timer T2 State. This status bit provides the current state of timer T2. 0 timer T2 is idle 1 timer T2 is running
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 T1_REP	Timer T1 Repetitive Mode. This bit configures the repetition mode of timer T1. 0 T1 is non repetitive 1 T1 is repetitive

Table continues on the next page...

FR_TICCR field descriptions (continued)

Field	Description
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 T1SP	Timer T1 Stop. This trigger bit is used to stop timer T1. 0 no effect 1 stop timer T1
14 T1TR	Timer T1 Trigger. This trigger bit is used to start timer T1. 0 no effect 1 start timer T1
15 T1ST	Timer T1 State. This status bit provides the current state of timer T1. 0 timer T1 is idle 1 timer T1 is running

54.6.41 Timer 1 Cycle Set Register (FR_TI1CYSR)

Write: Anytime

This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_CYC_VAL						0		T1_CYC_MSK					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1CYSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T1_CYC_VAL	Timer T1 Cycle Filter Value. This field defines the cycle filter value for timer T1.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T1_CYC_MSK	Timer T1 Cycle Filter Mask. This field defines the cycle filter mask for timer T1.

54.6.42 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

Write: Anytime

This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Eh offset = 5Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_MTOFFSET													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1MTOR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T1_MTOFFSET	Timer 1 Macrotick Offset. This field defines the macrotick offset value for timer 1.

54.6.43 Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, .

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2CYCVAL						0		T2CYCMSK					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T2CYCVAL	Timer T2 Cycle Filter Mask
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T2CYCMSK	Timer T2 Cycle Filter Mask

54.6.44 Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	T2MTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick High Word

54.6.45 Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)

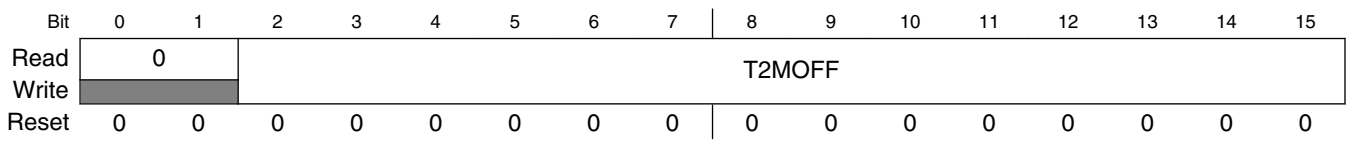
Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Absolute Timer T2

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.

Address: 0h base + 62h offset = 62h



FR_TI2CR1_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T2MOFF	Timer T2 Macrotick Offset

54.6.46 Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)

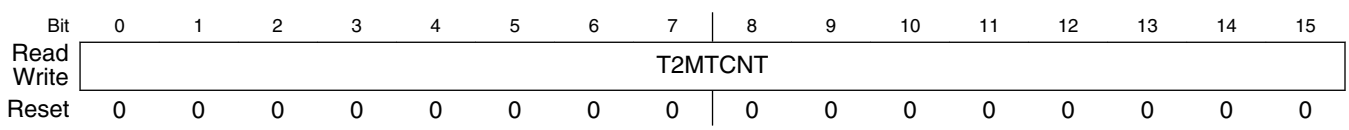
Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 62h offset = 62h



FR_TI2CR1_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick Low Word

54.6.47 Slot Status Selection Register (FR_SSSR)

Write: Anytime

This register is used to access the four internal non-memory mapped slot status selection registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding Slot Status Registers (FR_SSR0-FR_SSR7) according to Slot Status Selection Register (FR_SSSR) . For a detailed description of slot status monitoring, refer to Slot Status Monitoring .

Table 54-10. Mapping Between FR_SSSRn and FR_SSRn

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR0	FR_SSR0[15:8]	FR_SSR0[7:0]	FR_SSR1[15:8]	FR_SSR1[7:0]
FR_SSSR1	FR_SSR2[15:8]	FR_SSR2[7:0]	FR_SSR3[15:8]	FR_SSR3[7:0]
FR_SSSR2	FR_SSR4[15:8]	FR_SSR4[7:0]	FR_SSR5[15:8]	FR_SSR5[7:0]
FR_SSSR3	FR_SSR6[15:8]	FR_SSR6[7:0]	FR_SSR7[15:8]	FR_SSR7[7:0]

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	SLOTNUMBER										
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSSR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_SSSR field descriptions (continued)

Field	Description
2–3 SEL	Selector. This field selects one of the four internal slot status selection registers for access. 00 select FR_SSSR0. 01 select FR_SSSR1. 10 select FR_SSSR2. 11 select FR_SSSR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTNUMBER	Slot Number. This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. **NOTE:** If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.

54.6.48 Slot Status Counter Condition Register (FR_SSCCR)

Write: Anytime

This register is used to access and program the four internal non-memory mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal slot status counter condition registers defines the mode and the conditions for incrementing the counter in the corresponding Slot Status Counter Registers (FR_SSCR0-FR_SSCR3). The correspondence is given in Slot Status Counter Condition Register (FR_SSCCR) . For a detailed description of slot status counters, refer to Slot Status Counter Registers .

Table 54-11. Mapping between internal FR_SSCCRn and FR_SSCRn

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

Address: 0h base + 66h offset = 66h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK			
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SEL	Selector. This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0. 01 select FR_SSCCR1. 10 select FR_SSCCR2. 11 select FR_SSCCR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–6 CNTCFG	Counter Configuration. These bit field controls the channel related incrementing of the slot status counter. 00 increment by 1 if condition is fulfilled on channel A. 01 increment by 1 if condition is fulfilled on channel B. 10 increment by 1 if condition is fulfilled on at least one channel. 11 increment by 2 if condition is fulfilled on both channels channel; increment by 1 if condition is fulfilled on only one channel.
7 MCY	Multi Cycle Selection. This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only. 1 The Slot Status Counter accumulates over multiple communication cycles.
8 VFR	Valid Frame Restriction. This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only. 1 The counter is restricted to valid frames only.
9 SYF	Sync Frame Restriction. This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit. 1 The counter is restricted to frames with the sync frame indicator bit set to 1.
10 NUF	Null Frame Restriction. This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit. 1 The counter is restricted to frames with the null frame indicator bit set to 0.
11 SUF	Startup Frame Restriction. This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit. 1 The counter is restricted to received frames with the startup frame indicator bit set to 1.
12–15 STATUSMASK	Slot Status Mask. This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3] - This bit enables the counting for slots with the syntax error indicator bit set to 1. STATUSMASK[2] - This bit enables the counting for slots with the content error indicator bit set to 1.

Table continues on the next page...

FR_SSCCR field descriptions (continued)

Field	Description
	STATUSMASK[1] - This bit enables the counting for slots with the boundary violation indicator bit set to 1. STATUSMASK[0] - This bit enables the counting for slots with the transmission conflict indicator bit set to 1.

54.6.49 Slot Status Register (FR_SSR*n*)

Each Slot Status Register (SSR) holds the status vector of the slot specified in the corresponding internal slot status selection register, which can be programmed using the Slot Status Selection Register (FR_SSSR). Each register is updated after the end of the corresponding slot as shown in Figure 54-31 . The register bits are directly related to the protocol variables and described in more detail in Slot Status Monitoring .

Address: 0h base + 68h offset + (2d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSR*n* field descriptions

Field	Description
0 VFB	Valid Frame on Channel B. protocol related variable: vSS!ValidFrame channel B 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
1 SYB	Sync Frame Indicator Channel B. protocol related variable: vRF!Header!SyFIndicator channel B 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
2 NFB	Null Frame Indicator Channel B. protocol related variable: vRF!Header!NFIndicator channel B 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
3 SUB	Startup Frame Indicator Channel B. protocol related variable: vRF!Header!SuFIndicator channel B 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
4 SEB	Syntax Error on Channel B. protocol related variable: vSS!SyntaxError channel B 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1

Table continues on the next page...

FR_SSR*n* field descriptions (continued)

Field	Description
5 CEB	Content Error on Channel B. protocol related variable: vSS!ContentError channel B 0 vSS!ContentError = 0 1 vSS!ContentError = 1
6 BVB	Boundary Violation on Channel B. protocol related variable: vSS!BViolation channel B 0 vSS!BViolation = 0 1 vSS!BViolation = 1
7 TCB	Transmission Conflict on Channel B. protocol related variable: vSS!TxConflict channel B 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1
8 VFA	Valid Frame on Channel A. protocol related variable: vSS!ValidFrame channel A 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
9 SYA	Sync Frame Indicator Channel A. protocol related variable: vRF!Header!SyFIndicator channel A 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
10 NFA	Null Frame Indicator Channel A. protocol related variable: vRF!Header!NFIndicator channel A 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
11 SUA	Startup Frame Indicator Channel A. protocol related variable: vRF!Header!SuFIndicator channel A 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
12 SEA	Syntax Error on Channel A. protocol related variable: vSS!SyntaxError channel A 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1
13 CEA	Content Error on Channel A. protocol related variable: vSS!ContentError channel A 0 vSS!ContentError = 0 1 vSS!ContentError = 1
14 BVA	Boundary Violation on Channel A. protocol related variable: vSS!BViolation channel A 0 vSS!BViolation = 0 1 vSS!BViolation = 1
15 TCA	Transmission Conflict on Channel A. protocol related variable: vSS!TxConflict channel A 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1

54.6.50 Slot Status Counter Register (FR_SSCR*n*)

Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal slot status counter condition register FR_SSCCRn, which can be programmed by using the Slot Status Counter Condition Register (FR_SSCCR). For more details, see Slot Status Counter Registers .

NOTE

If the counter has reached its maximum value 0xFFFF and is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be set into the multicycle mode again.

NOTE

Address: 0h base + 78h offset + (2d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SLOTSTATUSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCR*n* field descriptions

Field	Description
0–15 SLOTSTATUSCNT	Slot Status Counter. This field provides the current value of the Slot Status Counter.

54.6.51 MTS A Configuration Register (FR_MTSACFR)

Write: MTE: Anytime; CYCCNTMSK,CYCCNTVAL: POC:config

This register controls the transmission of the Media Access Test Symbol MTS on channel A. For more details, see MTS Generation .

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSACFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

54.6.52 MTS B Configuration Register (FR_MTSBCFR)

Write: MTE: Anytime; CYCCNTMSK,CYCCNTVAL: POC:config

This register controls the transmission of the Media Access Test Symbol MTS on channel B. For more details, see MTS Generation .

Address: 0h base + 82h offset = 82h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSBCFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

54.6.53 Receive Shadow Buffer Index Register (FR_RSBIR)

Write: WMD, SEL: Any Time; RSBIDX: POC:config

This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers . For more details on the receive shadow buffer concept, refer to Receive Shadow Buffers Concept .

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7
Read Write	WMD	Reserved	SEL		Reserved			
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	RSBIDX							
Reset	0	0	0	0	0	0	0	0

FR_RSBIR field descriptions

Field	Description
0 WMD	Write Mode. This bit controls the write mode for this register. 0 update SEL and RSBIDX field on register write 1 update only SEL field on register write
1 Reserved	This field is reserved.
2–3 SEL	Selector. This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1. receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2. receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1. receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2. receive shadow buffer index register for channel B, segment 2
4–7 Reserved	This field is reserved.
8–15 RSBIDX	RSBIDXA1/RSBIDXA2/RSBIDXB1/RSBIDXB2- Receive Shadow Buffer Index Receive Shadow Buffer Index. This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation.The application provides initial message buffer header index value in the configuration phase. CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are 0 <= i <= 131 . Illegal values will be detected during the message buffer search.

54.6.54 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Write: WMA/WMB: POC:config, SEL: Anytime

This register is used to

- select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in Receive FIFO Watermark and Selection Register (FR_RFWMSR) .
- to define the watermark for the selected FIFO.

Table 54-12. SEL Controlled Receiver FIFO Registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Address: 0h base + 86h offset = 86h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	WM								0							SEL
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFWMSR field descriptions

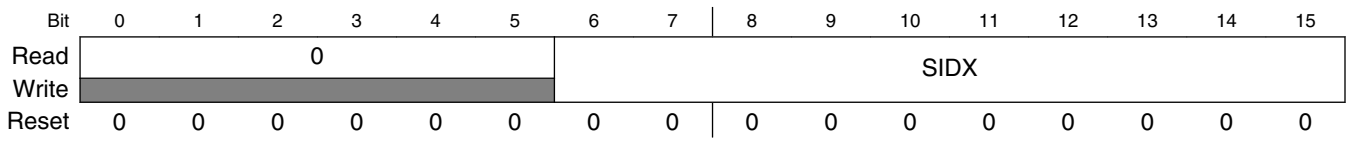
Field	Description
0–7 WM	WMA/WMB - Watermark This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 SEL	Select. This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

54.6.55 Receive FIFO Start Index Register (FR_RFSIR)

Write: POC:config

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Address: 0h base + 88h offset = 88h



FR_RFSIR field descriptions

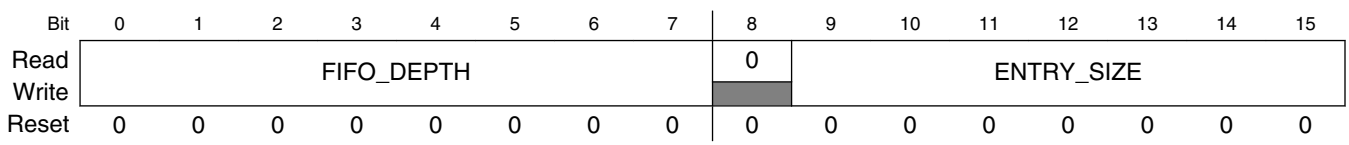
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SIDX	SIDXA/SIDXB - Start Index This field defines the number of the message buffer header field of the first message buffer of the selected FIFO. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

54.6.56 Receive FIFO Depth and Size Register (FR_RFDSR)

Write: POC:config

This register defines the structure of the selected FIFO, i.e. the number of entries and the size of each entry .

Address: 0h base + 8Ah offset = 8Ah



FR_RFDSR field descriptions

Field	Description
0–7 FIFO_DEPTH	FIFO_DEPTHA/FIFO_DEPTHB - FIFO Depth FIFO Depth. This field defines the depth of the selected FIFO, i.e. the number of entries. Note: If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 too, to ensure that no frames are received into the FIFO.

Table continues on the next page...

FR_RFDSR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 ENTRY_SIZE	ENTRY_SIZEA/ENTRY_SIZEB - Entry Size This field defines the size of the frame data sections for the selected FIFO in 2 byte entities.

54.6.57 Receive FIFO A Read Index Register (FR_RFARIR)

This register provides the message buffer header index of the next available FIFO A entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						RDIDX									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFARIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

54.6.58 Receive FIFO B Read Index Register (FR_RFBRIR)

This register provides the message buffer header index of the next available FIFO B entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Eh offset = 8Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								RDIDX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFBRIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

54.6.59 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Write: POC:config

This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFVR field descriptions

Field	Description
0–15 MIDAFVAL	MIDAFVALA/MIDAFVALB - Message ID Acceptance Filter Value Filter value for the message ID acceptance filter.

54.6.60 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

Write: POC:config

This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 92h offset = 92h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFMSK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFMR field descriptions

Field	Description
0–15 MIDAFMSK	MIDAFMSKA/MIDAFMSKB - Message ID Acceptance Filter Mask Filter mask for the message ID acceptance filter.

54.6.61 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

Write: POC:config

This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFVAL										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFVR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFVAL	FIDRFVALA/FIDRFVALB - Frame ID Rejection Filter Value Filter value for the frame ID rejection filter.

54.6.62 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Write: POC:config

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 96h offset = 96h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFMSK										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFMR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFMSK	Frame ID Rejection Filter Mask. Filter mask for the frame ID rejection filter.

54.6.63 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Write: WMD, IBD, SEL: Any Time; SID: POC:config

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter see FIFO Filtering .

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	IBD	SEL		0	SID										
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFRFCFR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL and IBD field only on write access.
1 IBD	Interval Boundary. This control bit selects the interval boundary to be programmed with the SID value. 0 program lower interval boundary 1 program upper interval boundary
2–3 SEL	Filter Selector. This control field selects the frame ID range filter to be accessed. 00 select frame ID range filter 0. 01 select frame ID range filter 1. 10 select frame ID range filter 2. 11 select frame ID range filter 3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFRFCFR field descriptions (continued)

Field	Description
5–15 SID	Slot ID Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

54.6.64 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Write: Anytime

This register is used to enable and disable each frame ID range filter and to define whether it is running as acceptance or rejection filter.

Address: 0h base + 9Ah offset = 9Ah

Bit	0	1	2	3	4	5	6	7
Read	0				F3MD	F2MD	F1MD	F0MD
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				F3EN	F2EN	F1EN	F0EN
Write								
Reset	0	0	0	0	0	0	0	0

FR_RFRFCTR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 F3MD	Range Filter 3 Mode. This control bit defines the filter mode of the frame ID range filter 3. 0 range filter 3 runs as acceptance filter 1 range filter 3 runs as rejection filter
5 F2MD	Range Filter 2 Mode. This control bit defines the filter mode of the frame ID range filter 2. 0 range filter 2 runs as acceptance filter 1 range filter 2 runs as rejection filter
6 F1MD	Range Filter 1 Mode. This control bit defines the filter mode of the frame ID range filter 1. 0 range filter 1 runs as acceptance filter 1 range filter 1 runs as rejection filter
7 F0MD	Range Filter 0 Mode. This control bit defines the filter mode of the frame ID range filter 0. 0 range filter 0 runs as acceptance filter 1 range filter 0 runs as rejection filter
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFRFCTR field descriptions (continued)

Field	Description
12 F3EN	Range Filter 3 Enable. This control bit is used to enable and disable the frame ID range filter 3. 0 range filter 3 disabled 1 range filter 3 enabled
13 F2EN	Range Filter 2 Enable. This control bit is used to enable and disable the frame ID range filter 2. 0 range filter 2 disabled 1 range filter 2 enabled
14 F1EN	Range Filter 1 Enable. This control bit is used to enable and disable the frame ID range filter 1. 0 range filter 1 disabled 1 range filter 1 enabled
15 F0EN	Range Filter 0 Enable. This control bit is used to enable and disable the frame ID range filter 0. 0 range filter 0 disabled 1 range filter 0 enabled

54.6.65 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTA	Last Dynamic Transmission Slot Channel A. protocol related variable: zLastDynTxSlot channel A Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

54.6.66 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Eh offset = 9Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTB	Last Dynamic Transmission Slot Channel B. protocol related variable: zLastDynTxSlot channel B Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

54.6.67 Protocol Configuration Register 0 (FR_PCR0)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Table 54-13. Protocol Configuration Register (PCR0-30) Fields

Name	Description	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts			number	3
action_point_offset	gdActionPointOffset - 1			MT	0
cas_rx_low_max	gdCASRxLowMax - 1			gdBit	4
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase			minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1			MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1			MT	2
static_slot_length	gdStaticSlot			MT	0

Table continues on the next page...

Table 54-13. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1			MT	13
symbol_window_exists	gdSymbolWindow!=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1			MT	6
tss_transmitter	gdTSSTransmitter			gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle			gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow			gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow			gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle			gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow			gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1			µT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]			MT	6
macro_initial_offset_b	pMacroInitialOffset[B]			MT	16
macro_per_cycle	gMacroPerCycle			MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot			MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart			MT	28
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal			cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive			cyclepairs	8
minislot_exists	gNumberOfMinislots!=0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1			minislot	29
number_of_static_slots	gNumberOfStaticSlots			static slot	2
offset_correction_start	gOffsetCorrectionStart			MT	11
payload_length_static	gPayloadLengthStatic			2 bytes	19
max_payload_length_dynamic	pPayloadLengthDynMax			2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1			MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock			bool	26
allow_passive_to_active	pAllowPassiveToActive			cyclepairs	12
cluster_drift_damping	pClusterDriftDamping			µT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]			µT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]			µT	26
listen_timeout	pdListenTimeout - 1			µT	14/15
key_slot_id	pKeySlotId			number	18
key_slot_used_for_startup	pKeySlotUsedForStartup			bool	11
key_slot_used_for_sync	pKeySlotUsedForSync			bool	11
latest_tx	gNumberOfMinislots - pLatestTx			minislot	21
sync_node_max	gSyncNodeMax			number	30
micro_initial_offset_a	pMicroInitialOffset[A]			µT	20

Table continues on the next page...

Table 54-13. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
micro_initial_offset_b	pMicroInitialOffset[B]			µT	20
micro_per_cycle	pMicroPerCycle			µT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift			µT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift			µT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)			µT	7
offset_correction_out	pOffsetCorrectionOut			µT	9
rate_correction_out	pRateCorrectionOut			µT	14
single_slot_enabled	pSingleSlotEnabled			bool	10
wakeup_channel	pWakeupChannel	see Protocol Configuration Register 0 (FR_PCR0)			10
wakeup_pattern	pWakeupPattern			number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2			µT	19
decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2			µT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection			µT	29
extern_rate_correction	pExternRateCorrection			µT	21

Table 54-14. Wakeup Channel Selection

wakeup_channel	Wakeup Channel
0	A
1	B

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	action_point_offset						static_slot_length									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR0 field descriptions

Field	Description
0–5 action_point_offset	gdActionPointOffset - 1
6–15 static_slot_length	gdStaticSlot

54.6.68 Protocol Configuration Register 1 (FR_PCR1)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A2h offset = A2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		macro_after_first_static_slot													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 macro_after_ first_static_slot	gMacroPerCycle - gdStaticSlot

54.6.69 Protocol Configuration Register 2 (FR_PCR2)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	minislot_after_action_point						number_of_static_slots									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR2 field descriptions

Field	Description
0–5 minislot_after_ action_point	gdMinislot - gdMinislotActionPointOffset - 1
6–15 number_of_ static_slots	gNumberOfStaticSlots

54.6.70 Protocol Configuration Register 3 (FR_PCR3)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A6h offset = A6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_symbol_rx_low						minislot_action_point_offset					coldstart_attempts				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR3 field descriptions

Field	Description
0–5 wakeup_symbol_rx_low	gdWakeupSymbolRxLow
6–10 minislot_action_point_offset	gdMinislotActionPointOffset - 1
11–15 coldstart_attempts	gColdstartAttempts

54.6.71 Protocol Configuration Register 4 (FR_PCR4)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cas_rx_low_max							wakeup_symbol_rx_window								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR4 field descriptions

Field	Description
0–6 cas_rx_low_max	gdCASRxLowMax - 1

Table continues on the next page...

FR_PCR4 field descriptions (continued)

Field	Description
7–15 wakeup_symbol_ rx_window	gdWakeupSymbolRxWindow

54.6.72 Protocol Configuration Register 5 (FR_PCR5)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AAh offset = AAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	tss_transmitter				wakeup_symbol_tx_low						wakeup_symbol_rx_idle					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR5 field descriptions

Field	Description
0–3 tss_transmitter	gdTSSTransmitter
4–9 wakeup_symbol_ tx_low	gdWakeupSymbolTxLow
10–15 wakeup_symbol_ rx_idle	gdWakeupSymbolRxIdle

54.6.73 Protocol Configuration Register 6 (FR_PCR6)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	symbol_window_after_action_point							macro_initial_offset_a							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR6 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–8 symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1
9–15 macro_initial_offset_a	pMacroInitialOffset[A]

54.6.74 Protocol Configuration Register 7 (FR_PCR7)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AEh offset = AEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_b								micro_per_macro_nom_half							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR7 field descriptions

Field	Description
0–8 decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2
9–15 micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)

54.6.75 Protocol Configuration Register 8 (FR_PCR8)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	max_without_clock_correction_fatal				max_without_clock_correction_passive				wakeup_symbol_tx_idle							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR8 field descriptions

Field	Description
0–3 max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal
4–7 max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive
8–15 wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle

54.6.76 Protocol Configuration Register 9 (FR_PCR9)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B2h offset = B2h

Bit	0	1	2	3	4	5	6	7
Read Write	minislot_exists	symbol_window_exists	offset_correction_out					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_out							
Reset	0	0	0	0	0	0	0	0

FR_PCR9 field descriptions

Field	Description
0 minislot_exists	gNumberOfMinislots!=0
1 symbol_window_exists	gdSymbolWindow!=0

Table continues on the next page...

FR_PCR9 field descriptions (continued)

Field	Description
2–15 offset_ correction_out	pOffsetCorrectionOut

54.6.77 Protocol Configuration Register 10 (FR_PCR10)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7
Read Write	single_slot_ enabled	wakeup_ channel	macro_per_cycle					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	macro_per_cycle							
Reset	0	0	0	0	0	0	0	0

FR_PCR10 field descriptions

Field	Description
0 single_slot_ enabled	pSingleSlotEnabled
1 wakeup_channel	pWakeupChannel
2–15 macro_per_cycle	gMacroPerCycle

54.6.78 Protocol Configuration Register 11 (FR_PCR11)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B6h offset = B6h

Bit	0	1	2	3	4	5	6	7
Read Write	key_slot_used_for_startup	key_slot_used_for_sync	offset_correction_start					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_start							
Reset	0	0	0	0	0	0	0	0

FR_PCR11 field descriptions

Field	Description
0 key_slot_used_for_startup	pKeySlotUsedForStartup
1 key_slot_used_for_sync	pKeySlotUsedForSync
2–15 offset_correction_start	gOffsetCorrectionStart

54.6.79 Protocol Configuration Register 12 (FR_PCR12)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	allow_passive_to_active					key_slot_header_crc										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR12 field descriptions

Field	Description
0–4 allow_passive_to_active	pAllowPassiveToActive
5–15 key_slot_header_crc	header CRC for key slot

54.6.80 Protocol Configuration Register 13 (FR_PCR13)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BAh offset = BAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	first_minislot_action_point_offset						static_slot_after_action_point									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR13 field descriptions

Field	Description
0–5 first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1
6–15 static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1

54.6.81 Protocol Configuration Register 14 (FR_PCR14)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	rate_correction_out											listen_timeout				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR14 field descriptions

Field	Description
0–10 rate_correction_out	pRateCorrectionOut
11–15 listen_timeout	pdListenTimeout - 1

54.6.82 Protocol Configuration Register 15 (FR_PCR15)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BEh offset = BEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR15 field descriptions

Field	Description
0–15 listen_timeout	pdListenTimeout - 1

54.6.83 Protocol Configuration Register 16 (FR_PCR16)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	macro_initial_offset_b							noise_listen_timeout								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR16 field descriptions

Field	Description
0–6 macro_initial_ offset_b	pMacroInitialOffset[B]
7–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

54.6.84 Protocol Configuration Register 17 (FR_PCR17)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	noise_listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR17 field descriptions

Field	Description
0–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

54.6.85 Protocol Configuration Register 18 (FR_PCR18)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_pattern						key_slot_id									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR18 field descriptions

Field	Description
0–5 wakeup_pattern	pWakeupPattern
6–15 key_slot_id	pKeySlotId

54.6.86 Protocol Configuration Register 19 (FR_PCR19)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C6h offset = C6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_a								payload_length_static							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR19 field descriptions

Field	Description
0–8 decoding_ correction_a	pDecodingCorrection + pDelayCompensation[A] + 2
9–15 payload_length_ static	gPayloadLengthStatic

54.6.87 Protocol Configuration Register 20 (FR_PCR20)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_initial_offset_b								micro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR20 field descriptions

Field	Description
0–7 micro_initial_ offset_b	pMicroInitialOffset[A]
8–15 micro_initial_ offset_a	pMicroInitialOffset[A]

54.6.88 Protocol Configuration Register 21 (FR_PCR21)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	extern_rate_ correction			latest_tx												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR21 field descriptions

Field	Description
0–2 extern_rate_ correction	pExternRateCorrection
3–15 latest_tx	gNumberOfMinislots - pLatestTx

54.6.89 Protocol Configuration Register 22 (FR_PCR22)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved	comp_accepted_startup_range_a						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_a				micro_per_cycle			
Reset	0	0	0	0	0	0	0	0

FR_PCR22 field descriptions

Field	Description
0 Reserved	Reserved bit, will not be changed. Application must not write any value different from the reset value.

Table continues on the next page...

FR_PCR22 field descriptions (continued)

Field	Description
	This field is reserved.
1–11 comp_accepted_ startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]
12–15 micro_per_cycle	pMicroPerCycle

54.6.90 Protocol Configuration Register 23 (FR_PCR23)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CEh offset = CEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR23 field descriptions

Field	Description
0–15 micro_per_cycle	pMicroPerCycle

54.6.91 Protocol Configuration Register 24 (FR_PCR24)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cluster_drift_damping					max_payload_length_dynamic							micro_per_cycle_min			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR24 field descriptions

Field	Description
0–4 cluster_drift_ damping	pClusterDriftDamping
5–11 max_payload_ length_dynamic	pPayloadLengthDynMax
12–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

54.6.92 Protocol Configuration Register 25 (FR_PCR25)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D2h offset = D2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_min															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR25 field descriptions

Field	Description
0–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

54.6.93 Protocol Configuration Register 26 (FR_PCR26)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7
Read Write	allow_halt_due_to_clock	comp_accepted_startup_range_b						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_b				micro_per_cycle_max			
Reset	0	0	0	0	0	0	0	0

FR_PCR26 field descriptions

Field	Description
0 allow_halt_due_to_clock	pAllowHaltDueToClock
1–11 comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]
12–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

54.6.94 Protocol Configuration Register 27 (FR_PCR27)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D6h offset = D6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_max															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR27 field descriptions

Field	Description
0–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

54.6.95 Protocol Configuration Register 28 (FR_PCR28)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	dynamic_slot_idle_phase		macro_after_offset_correction													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR28 field descriptions

Field	Description
0–1 dynamic_slot_idle_phase	gdDynamicSlotIdlePhase.
2–15 macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart

54.6.96 Protocol Configuration Register 29 (FR_PCR29)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DAh offset = DAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	extern_offset_correction			minislots_max												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR29 field descriptions

Field	Description
0–2 extern_offset_correction	pExternOffsetCorrection

Table continues on the next page...

FR_PCR29 field descriptions (continued)

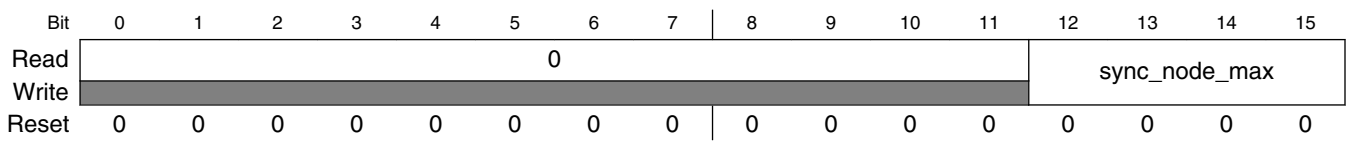
Field	Description
3–15 minislots_max	gNumberOfMinislots - 1

54.6.97 Protocol Configuration Register 30 (FR_PCR30)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DCh offset = DCh

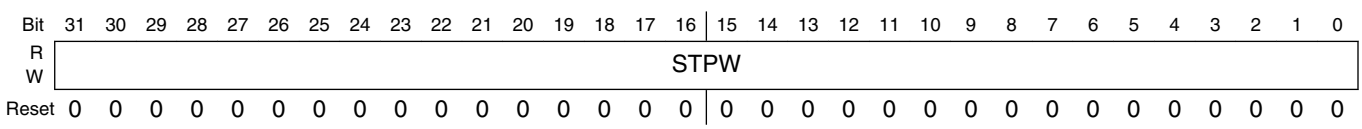


FR_PCR30 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 sync_node_max	gSyncNodeMax

54.6.98 StopWatch Count Register (FR_STPWR)

Address: 0h base + DEh offset = DEh



FR_STPWR field descriptions

Field	Description
STPW	StopWatch Count Register This is the value of the stopwatch counter running on system clock. This starts on cycle start even and stops on an external event “OS_tick”. This is a read reset type register, which means that when all the 32 bits are read, it automatically clears. For details, please see the "StopWatch Function" section.

54.6.99 Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

Write: Anytime

This register defines whether or not the Event output ports are enabled based on the events such as Cycle start and Timer1 and Timer2 expiry (see Timer Support for details).

Address: 0h base + E2h offset = E2h

Bit	0	1	2	3	4	5	6	7
Read	1							STPW_EN
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0					TIM2_EE	TIM1_EE	CYC_EE
Write								
Reset	0	0	0	0	0	0	0	0

FR_PEOER field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
7 STPW_EN	Stopwatch count Enable This bit controls the stopwatch counter. 0 Stopwatch counter disabled 1 Stopwatch counter enabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 TIM2_EE	Timer 2 expired Event Output Enable- This bit controls the event on port "fr_evt_tim2" 0 Timer 2 expired event out disabled 1 Timer 2 expired event out enabled
14 TIM1_EE	Timer 1 expired Event Output Enable- This bit controls the event on port "fr_evt_tim1" 0 Timer 1 expired event out disabled 1 Timer 1 expired event out enabled
15 CYC_EE	Cycle Start Event Output Enable- This bit controls the event on port "fr_evt_cyc" 0 Cycle start event out disabled 1 Cycle start event out enabled

54.6.100 Receive FIFO Start Data Offset Register (FR_RFSDOR)

Write: POC:config

NOTE

Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses the content of the FR_RFSDOR corresponds to the start address of payload area of the selected FIFO.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSDOR field descriptions

Field	Description
0–15 SDO	SDOA/SDOB - Start Data Field Offset Start Data Field Offset. This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO's first message buffer header field. For configuration constraints see Configure Data Field Offsets .

54.6.101 Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Write: Disabled Mode

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

54.6.102 Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Write: Disabled Mode

Address: 0h base + EAh offset = EAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SMBA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.6.103 Receive FIFO Periodic Timer Register (FR_RFPTR)

Write: POC:config

This register holds periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (see FIFO Periodic Timer).

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		PTD													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFPTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFPTR field descriptions (continued)

Field	Description
2–15 PTD	Periodic Timer Duration. This value defines the periodic timer duration in terms of macroticks. 0000 timer stays expired 3FFF timer never expires other timer expires after specified number of macroticks, expires and is restarted at each cycle start

54.6.104 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs

If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, than the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

.

Address: 0h base + EEh offset = EEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	FLB_or_PCB								FLA_or_PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFLPCR field descriptions

Field	Description
0–7 FLB_or_PCB	**NOTE:** The name and function of the fields in this register vary depending on whether they are being read or written. See the following table for details.

Operation	Sub-Field	Description
Read	0-7 FLB	Fill Level FIFO B. This field provides the current number of entries in the FIFO B.
	-	-
Write	0-7 PCB	Pop Count FIFO B. This field defines the number of entries to be removed from FIFO B.
	-	-

Field	Description
8–15 FLA_or_PCA	

Operation	Sub-Field	Description
Read	-	-

Table continues on the next page...

FR_RFFLPCR field descriptions (continued)

Field	Description		
	Operation	**Sub-Field**	**Description**
		8-15 FLA	Fill Level FIFO A- This field provides the current number of entries in the FIFO A.
	Write	-	-
		8-15 PCA	Pop Count FIFO A - This field defines the number of entries to be removed from FIFO A.

54.6.105 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

This register provides the means to control the ECC related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description see Memory Error Reporting .

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7
Read	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
Write								
Reset	0	0	0	0	0	0	0	0

FR_EEIFER field descriptions

Field	Description
0 LRNE_OF	LRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on CHI LRAM and interrupt flag LRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of CHI LRAM 0 no such event 1 Non-Corrected Error overflow detected on CHI LRAM
1 LRCE_OF	LRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears a) memory errors are detected and corrected on CHI LRAM and interrupt flag LRCE_IF is already 1.

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
	b) memory errors are detected and corrected on at least two banks of CHI LRAM **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error overflow detected on CHI LRAM
2 DRNE_OF	DRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on PE DRAM and interrupt flag DRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of the PE DRAM 0 no such event 1 Non-Corrected Error overflow detected on PE DRAM
3 DRCE_OF	DRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected and corrected on PE DRAM and interrupt flag DRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of PE DRAM 0 no such event 1 Corrected Error overflow detected on PE DRAM
4 LRNE_IF	LRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on the CHI LRAM. 0 no such event 1 Non-Corrected Error detected on CHI LRAM
5 LRCE_IF	LRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on the CHI LRAM. **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error detected on CHI LRAM
6 DRNE_IF	DRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on PE DRAM. 0 no such event 1 Non-Corrected Error detected on PE DRAM
7 DRCE_IF	DRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on PE DRAM. 0 no such event 1 Corrected Error detected on PE DRAM
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 LRNE_IE	LRAM Non-Corrected Error Interrupt Enable. This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
13 LRCE_IE	LRAM Corrected Error Interrupt Enable. This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 DRNE_IE	DRAM Non-Corrected Error Interrupt Enable. This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 DRCE_IE	DRAM Corrected Error Interrupt Enable. This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line

54.6.106 ECC Error Report and Injection Control Register (FR_EERICR)

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Address: 0h base + F2h offset = F2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	BSY	0					ERS		0			ERM	0		EIM	EIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERICR field descriptions

Field	Description
0 BSY	Register Update Busy- This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in " Register Write Access 0 ECC configuration is idle 1 ECC configuration is running
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 ERS	Error Report Select. This field selects the content of the ECC Error reporting registers. 00 show PE DRAM non-corrected error information 01 show PE DRAM corrected error information

Table continues on the next page...

FR_EERICR field descriptions (continued)

Field	Description
	10 show CHI LRAM non-corrected error information 11 show CHI LRAM corrected error information
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERM	Error Report Mode. This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 store data and code as delivered by ecc decoding logic. 1 store data and code as read from the memory.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 EIM	Error Injection Mode. This bit configures the ECC error injection mode. 0 use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection. 1 use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection.
15 EIE	Error Injection Enable. This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

54.6.107 ECC Error Report Address Register (FR_EERAR)

This register provides the memory identifier, bank, and address for which the memory error is reported.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MID	BANK			ADDR											
Write																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field provides the BANK for which the memory error is reported. 111 reset value, indicates no error found after reset. For MID=0:

Table continues on the next page...

FR_EERAR field descriptions (continued)

Field	Description
	000 PE DRAM [7:0] 001 PE DRAM [15:8] others - not used For MID=1: Refer to the following table for the assignment of the LRAM banks.

Table 54-15. LRAM Bank Value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

4–15 ADDR	Memory Address. This field provides the address of the failing memory location.

54.6.108 ECC Error Report Data Register (FR_EERDR)

This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

Table 54-16. Valid Bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0									PE DRAM[7:0]							
PE DRAM	1									PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1						FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2									FR_MBIDXR(2n)[MBIDX]							
CHI LRAM	2	FR_MBDOR(6n+2)															
CHI LRAM	3	FR_MBCCFR(2n+1)															

Table continues on the next page...

Table 54-16. Valid Bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field (continued)

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHI LRAM	3	FR_MBDOR(6n+3)															
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5									FR_MBIDXR(2n+1)[MBIDX]							
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

Address: 0h base + F6h offset = F6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Ecc Data, shows data as generated by the ecc decoding logic. ERM=1: Memory Data, shows data as read from the memory.

54.6.109 ECC Error Report Code Register (FR_EERCR)

This register provides the ECC related information of the reported memory read access.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERCR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_EERCR field descriptions (continued)

Field	Description
11–15 CODE	Code. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Syndrome. Shows the ecc syndrome generated by the ecc decoding logic. The coding of the PE DRAM syndrome is shown in PE DRAM Syndrome . The coding of the CHI LRAM syndrome is shown in CHI LRAM Syndrome . ERM=1: Checkbits. Shows the ecc checkbits read from the memory.

54.6.110 ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Address: 0h base + FAh offset = FAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MID	BANK			ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field defines the memory bank for ECC error injection. For MID=0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] others reserved For MID=1: Refer to ECC Error Report Address Register (FR_EERAR) for the assignment of the LRAM banks.
4–15 ADDR	Memory Address. This flag defines the memory address for ECC error injection.

54.6.111 ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

NOTE

The effect of the error injected depends from the LRAM content at the address accessed and from the module internal usage of the data. Refer to Memory Error Response for details.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the data written into the memory. EIM=1: This field defines the data to be written into the memory.

54.6.112 ECC Error Injection Code Register (FR_EEICR)

This register defines the ECC code distortion pattern for the error injection write.

Address: 0h base + FEh offset = FEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEICR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CODE	Code. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the ecc checkbits written into the memory. EIM=1: This field defines the ecc checkbits written into the memory.

54.6.113 Message Buffer Configuration, Control, Status Register (FR_MBCCSR*n*)

Write: MTD: POC:config or MB_DIS CMT: MB_LCK or MB_DIS EDT, LCKT, MBIE, MBIF: Normal Mode

The content of these registers is composed of message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in Individual Message Buffer Functional Description.

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Address: 0h base + 800h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7
Read	0			MTD	CMT		0	MBIE
Write						EDT	LCKT	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DUP	DVAL	EDS	LCKS	MBIF
Write								w1c
Reset	0	0	0	0	0	0	0	0

FR_MBCCSR*n* field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MTD	Message Buffer Transfer Direction. This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
4 CMT	Commit for Transmission. This bit indicates if the transmit message buffer data are ready for transmission. **NOTE:** This bit is read/write but may be modified by hardware other than by a reset. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
5 EDT	Enable/Disable Trigger. If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
6 LCKT	Lock/Unlock Trigger. If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit. 0 No effect 1 Message buffer lock or unlock is triggered
7 MBIE	Message Buffer Interrupt Enable. This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set.

Table continues on the next page...

FR_MBCCSR*n* field descriptions (continued)

Field	Description
	0 Interrupt request generation disabled 1 Interrupt request generation enabled
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 DUP	Data Updated. This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and Message buffer data field not updated 1 Frame Header and Message buffer data field updated
12 DVAL	Data Valid. For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 receive message buffer contains no valid frame data / message is transmitted for the first time 1 receive message buffer contains valid frame data / message will be transferred again
13 EDS	Enable/Disable Status. This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled. 1 Message buffer is enabled.
14 LCKS	Lock Status. This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application. 1 Message buffer is locked by the application.
15 MBIF	Message Buffer Interrupt Flag. This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled

54.6.114 Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to Message Buffer Cycle Counter Filtering .

NOTE

If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the message buffer configuration is illegal and the result of the message buffer search is not defined.

Table 54-17. Channel Assignment Description

CHA	CHB	Transmit Message Buffer		Receive Message Buffer	
		static segment	dynamic segment	static segment	dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Address: 0h base + 802h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7
Read Write	MTM	CHA	CHB	CCFE	CCFMSK			
Reset	0*	0*	0*	0*	0*	0*	0*	0*

Bit	8	9	10	11	12	13	14	15
Read Write	CCFMSK		CCFVAL					
Reset	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBCCFR*n* field descriptions

Field	Description
0 MTM	Message Buffer Transmission Mode. This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event transmission mode 1 State transmission mode
1 CHA	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .
2 CHB	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .

Table continues on the next page...

FR_MBCCFR*n* field descriptions (continued)

Field	Description
3 CCFE	Cycle Counter Filtering Enable. This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled
4–9 CCFMSK	Cycle Counter Filtering Mask. This field defines the filter mask for the cycle counter filtering.
10–15 CCFVAL	Cycle Counter Filtering Value. This field defines the filter value for the cycle counter filtering.

54.6.115 Message Buffer Frame ID Register (FR_MBFIDR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

Address: 0h base + 804h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FID										
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBFIDR*n* field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FID	Frame ID. The semantic of this field depends on the message buffer transfer type. • Receive Message Buffer: This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. • Transmit Message Buffer: This field is used to determine the slot in which the message in this message buffer should be transmitted.

54.6.116 Message Buffer Index Register (FR_MBIDXR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

Address: 0h base + 806h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								MBIDX							
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBIDXR*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 MBIDX	Message Buffer Index. This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are 0 <= i <= 131 . Illegal values will be detected during the message buffer search.

54.6.117 Message Buffer Data Field Offset Register (FR_MBDOR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config

Address: 0h base + 1000h offset + (2d × i), where i=0d to 131d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MBDO															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBDOR*n* field descriptions

Field	Description
0–15 MBDO	Message Buffer Data Field Offset. This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints see Configure Data Field Offsets .

54.6.118 LRAM ECC Error Test Register (FR_LEETR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 1108h offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	LEETD															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_LEETR*n* field descriptions

Field	Description
0–15 LEETD	LRAM ECC Error Test Data. This field contains the LRAM data belonging to the test register located in LRAM Bank n.

54.7 Functional Description

This section provides a detailed description of the functionality implemented in the CC.

54.7.1 Message Buffer Concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts, the *message buffer control data* and the *physical message buffer*. The message buffer control data are located in dedicated registers. The structure of the message buffer control data depends on the

message buffer type and is described in the Message Buffer Types section. The physical message buffer is located in the FlexRay memory area and is described in the next section.

54.7.2 Physical Message Buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area.The structure of a physical message buffer is depicted in the figure below.

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

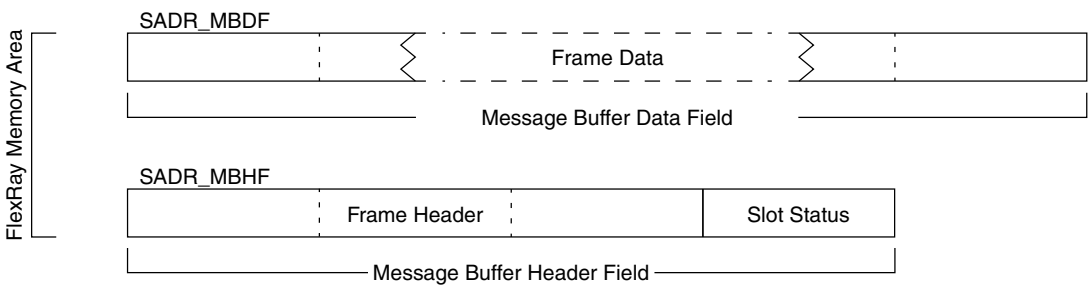


Figure 54-2. Physical Message Buffer Structure

54.7.2.1 Message Buffer Header Field

The message buffer header field is a contiguous region in the FlexRay memory area and occupies eight bytes. It contains the frame header, and the slot status. Its structure is shown in the above figure. The physical start address *SADR_MBHF* of the message buffer header field must be 16-bit aligned.

54.7.2.1.1 Frame Header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header related information according to the *FlexRay Communications System Protocol Specification, Version*2.1 Rev A. A detailed description of the usage and the content of the frame header is provided in Frame Header Description section.

54.7.2.1.2 Slot Status

The slot status occupies the last two bytes of the message buffer header field. It provides the slot and frame status related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in the Slot Status Description section.

54.7.2.2 Message Buffer Data Field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in the following section.

54.7.3 Message Buffer Types

The CC provides three different types of message buffers.

- *Individual Message Buffers*
- *Receive Shadow Buffers*
- *Receive FIFO Buffers*

For each message buffer type the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

54.7.3.1 Individual Message Buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in the Individual Message Buffer Functional Description section.

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in the figure below.

Each individual message buffer has a message buffer number n assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number *n* is controlled by the registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn.

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field MBIDX in the Message Buffer Index Registers (FR_MBIDXRn). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$SADR\_MBHF = (FR\_MBIDXRn[MBIDX] \times 8) + SMBA$$

Equation 32

The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i the FR_MBDORi contains the offset of the corresponding message buffer data field with respect to the CC FlexRay memory area base address as provided by SMBA field in the System Memory Base Address Register (FR_SYMBADR).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to this equation:

$$SADR\_MBDF = [DataFieldOffset] + SMBA$$

Equation 33

The FR_MBDORn are stored in the module internal memory LRAM. Refer to CHI LRAM Initialization for the setup of the data field offset values.

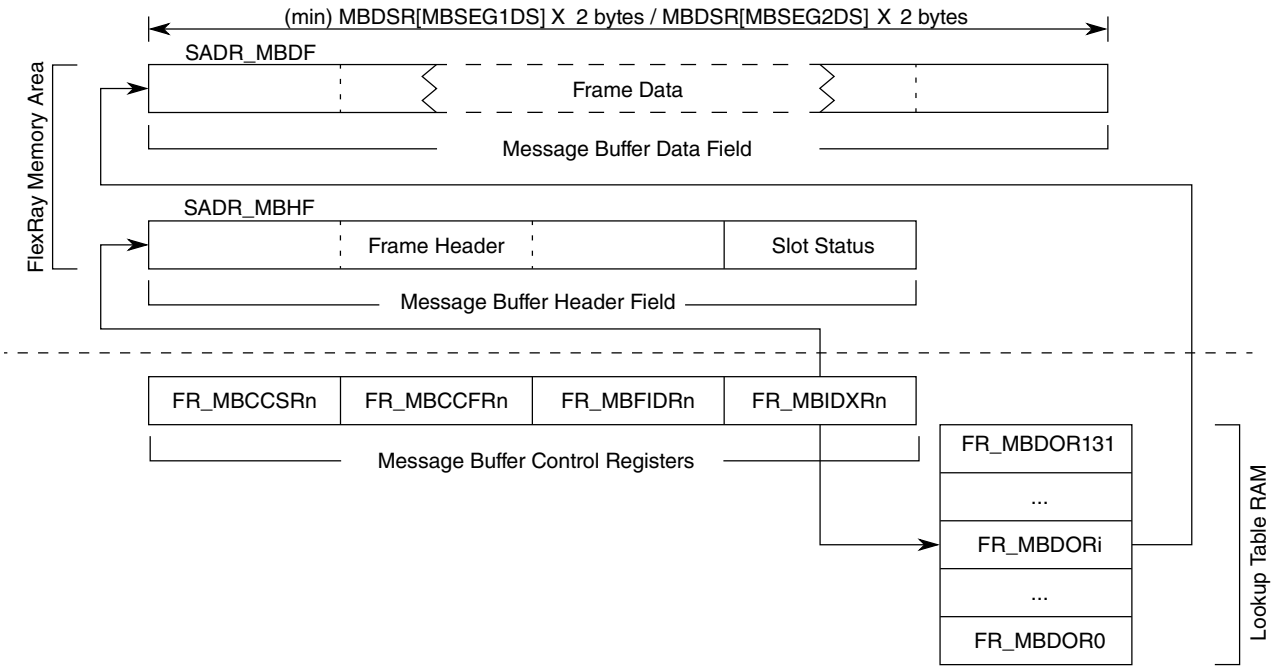


Figure 54-3. Individual Message Buffer Structure

54.7.3.1.1 Individual Message Buffer Segments

The set of the individual message buffers can be split up into two message buffer segments using the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). All individual message buffers with a message buffer number n ≤ FR_MBSSUTR[LAST_MB_SEG1] belong to the first message buffer segment. All individual message buffers with a message buffer number n > FR_MBSSUTR[LAST_MB_SEG1] belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- *all physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length*
- *the minimum length of the message buffer data field for individual message buffers in the first message buffer segment is 2 × FR_MBDSR[MBSEG1DS] bytes*
- *the minimum length of the message buffer data field for individual message buffers assigned to the second segment is 2 × FR_MBDSR[MBSEG2DS] bytes.*

54.7.3.2 Receive Shadow Buffers

The receive shadow buffers are required for the frame reception process for individual message buffers. The CC provides four receive shadow buffers, one receive shadow buffer per channel and per message buffer segment.

Each receive shadow buffer consists of two parts, the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in the figure below. The four internal shadow buffer control registers can be accessed by the Receive Shadow Buffer Index Register (FR_RSBIR).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the Receive Shadow Buffer Index Register (FR_RSBIR). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$SADR\_MBHF = (FR\_RSBIR[RSBIDX] \times 8) + SMBA$$

Equation 34

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in Receive Shadow Buffer Index Register (FR_RSBIR).

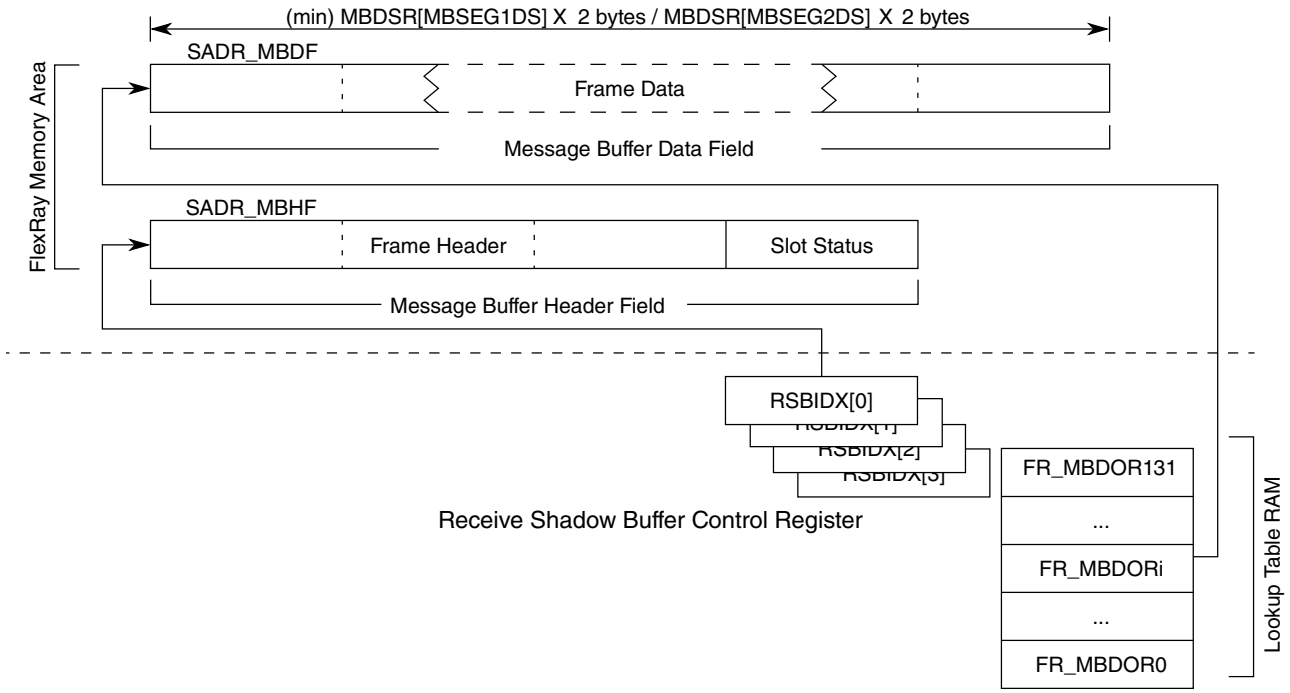


Figure 54-4. Receive Shadow Buffer Structure

54.7.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in the Receive FIFO Structure figure, later in this section.

The connection between the receive FIFO control registers and the set of physical message buffers is established by the Receive FIFO Start Index Register (FR_RFSIR), the Receive FIFO Depth and Size Register (FR_RFDSR), and the Receive FIFO A Read Index Register (FR_RFARIR) / Receive FIFO B Read Index Register (FR_RFBRIR).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM] in the Module Configuration Register.

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to Equation 35 on page 2586.

$$SADR\_MBHF[1] = (8 \times FR\_RFSIR[SIDX]) + SMBA$$

Equation 35

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according Equation 36 on page 2587 .

$$SADR\_MBHF[n] = (8 \times (FR\_RFSIR[SIDX] + FR\_RFDSR[FIFO\_DEPTH])) + SMBA$$

Equation 36

The required information to access the current entry of the FIFO is given in the following registers:

- *The registers Receive FIFO A Read Index Register (FR_RFARIR) and Receive FIFO B Read Index Register (FR_RFBRIR) provide the index of the physical message buffer belonging to the current entry.*

  The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

$$RF\_DFO[X] = FR\_RFSDOR[X] + (FR\_RFDSR[X][ENTRY\_SIZE] \times 2) \times i - FR\_RFSIDX[X])$$

Equation 37

Note

The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to FIFO Update for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

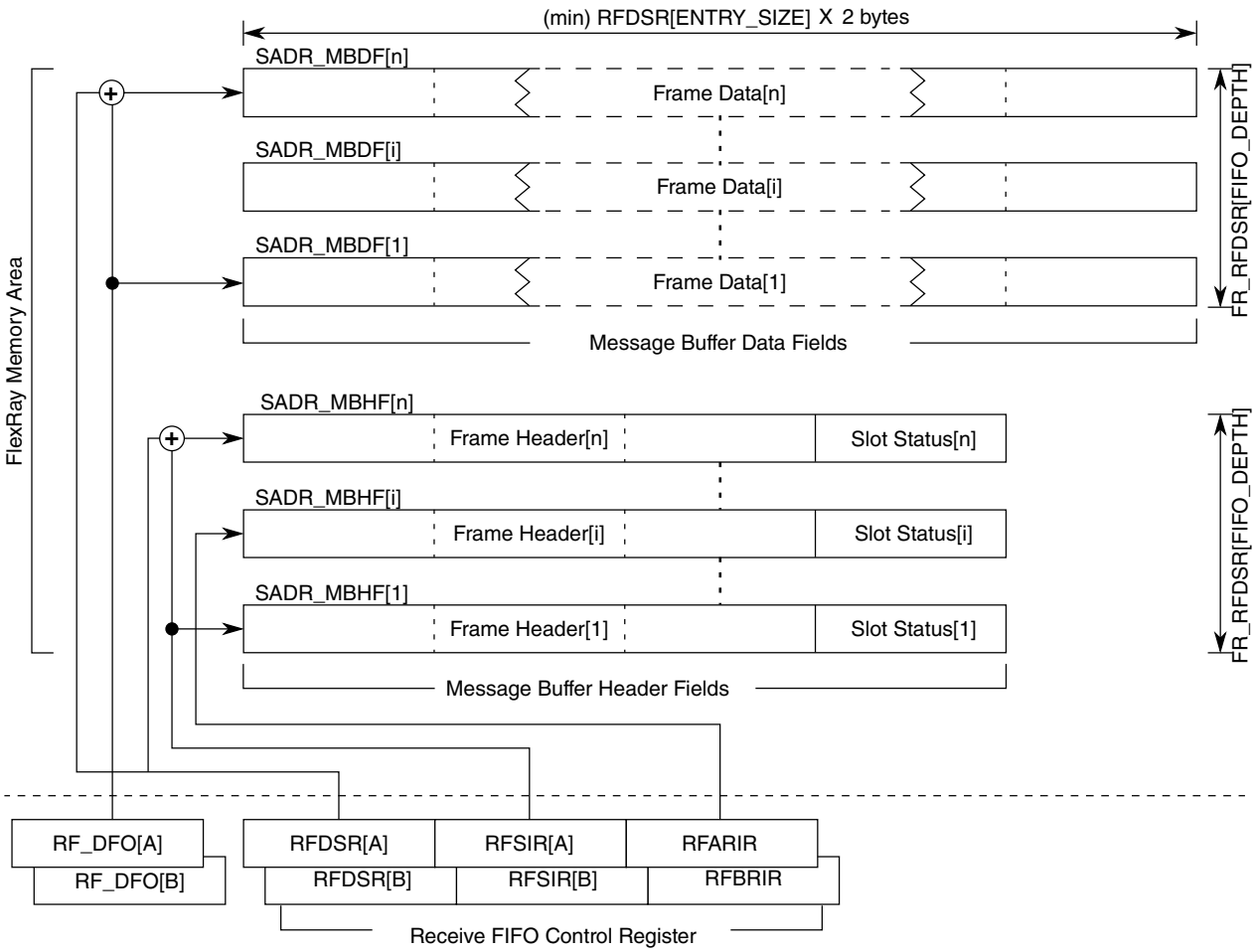


Figure 54-5. Receive FIFO Structure

Note

The actual values of the data field offsets RF_DFO[A/B] need to be calculated according to Equation 37 on page 2587. They are not stored in a register.

54.7.3.4 Message Buffer Configuration and Control Data

This section describes the configuration and control data for each message buffer type.

54.7.3.4.1 Individual Message Buffer Configuration Data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

54.7.3.4.1.1 Common Configuration Data

The set of common configuration data for individual message buffers is located in the following registers.

- *Message Buffer Data Size Register (FR_MBDSR)*

  The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.

- *Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)*

  The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are used. For more details, see Individual Message Buffer Segments.

54.7.3.4.1.2 Specific Configuration Data

The set of message buffer specific configuration data for individual message buffers is located in the following registers.

- *Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)*

  The MTD bit configures the message buffer type.

- *Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)*

  The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.

- *Message Buffer Frame ID Registers (FR_MBFIDRn)*

  For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.

- *Message Buffer Index Registers (FR_MBIDXRn)This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.*

54.7.3.5 Individual Message Buffer Control Data

During normal operation, each individual message buffer can be controlled by the control and trigger bits CMT, LCKT, EDT, and MBIE in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn).

54.7.3.6 Receive Shadow Buffer Configuration Data

Before frame reception into the individual message buffers can be performed, the receive shadow buffers must be configured. The configuration data are provided by the Receive Shadow Buffer Index Register (FR_RSBIR). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

54.7.3.7 Receive FIFO Control and Configuration Data

This section describes the configuration and control data for the two receive FIFOs.

54.7.3.7.1 Receive FIFO Configuration Data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- *Receive FIFO Periodic Timer Register (FR_RFPTR)*

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- *Receive FIFO Watermark and Selection Register (FR_RFWMSR)*
- *Receive FIFO Start Index Register (FR_RFSIR)*
- *Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *Receive FIFO Depth and Size Register (FR_RFDSR)*
- *Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)*
- *Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)*
- *Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)*

- *Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)*
- *Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)*

54.7.3.7.2 Receive FIFO Control Data

The application can access the FIFOs at any time using the control bits in the following registers:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

54.7.3.7.3 Receive FIFO Status Data

The current status of the receive fifo is provided in the following register:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO A Read Index Register (FR_RFARIR)*
- *Receive FIFO B Read Index Register (FR_RFBRIR)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

54.7.4 FlexRay Memory Area Layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

54.7.4.1 FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

In the figure given below, shows an example layout for the FIFO address mode FR_MCR[FAM]=0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is maximum 64 KB.
- The FlexRay memory area starts at a 16 byte boundary

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

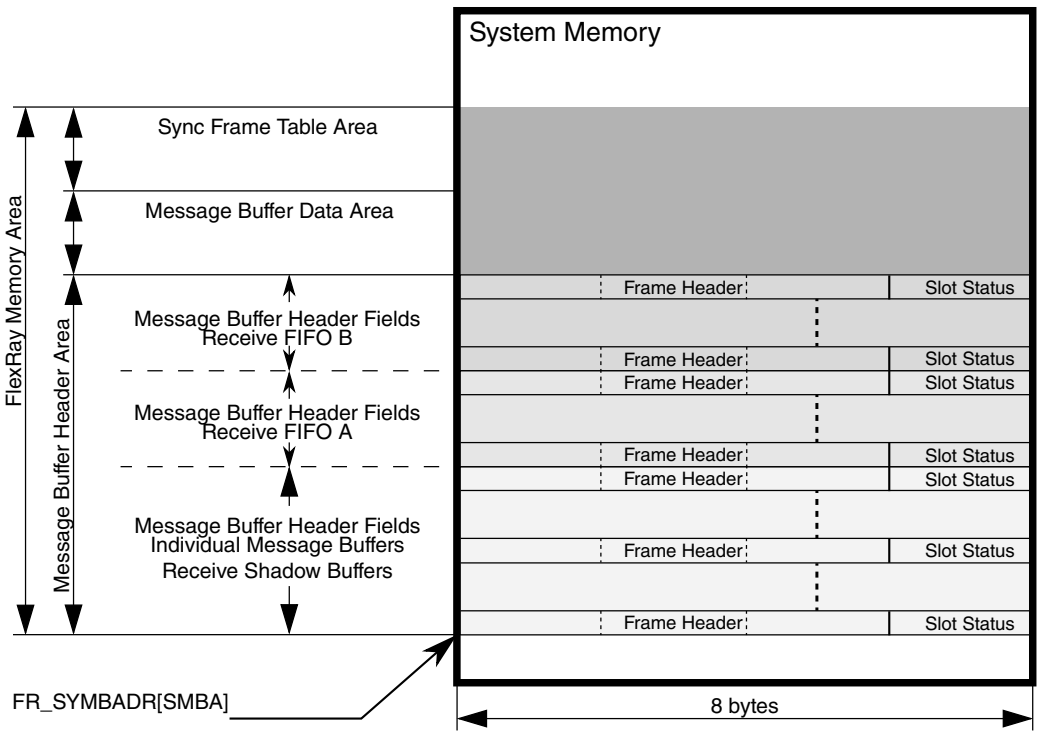


Figure 54-6. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

54.7.4.2 FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

The following figure shows an example layout for the FIFO address mode FR_MCR[FAM]=1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

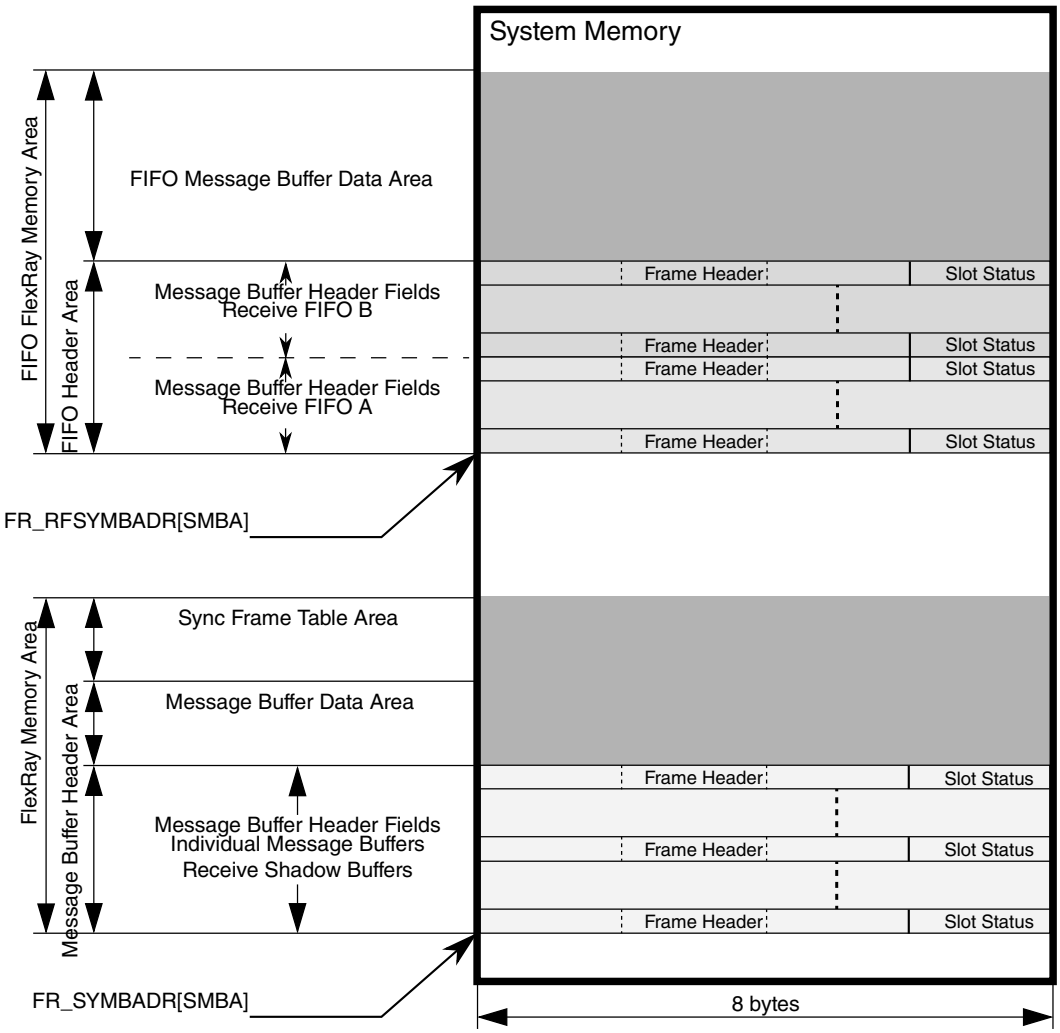


Figure 54-7. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

54.7.4.3 Message Buffer Header Area (FR_MCR[FAM] = 0)

The message buffer header area contains all message buffer header fields of the physical message buffers for all message buffer types. The following rules apply to the message buffer header fields for the three type of message buffers.

1. The start byte address SADR_MBHF of each message buffer header field for *individual message buffers* and *receive shadow buffers* must fulfill Equation 38 on page 2593.

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 131)$$

Equation 38

2. The start byte address SADR_MBHF of each message buffer header field for the *FIFO* must fulfill Equation 38 on page 2593.

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 39

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 40

3. The message buffer header fields for each FIFO have to be a contiguous area.

54.7.4.4 Message Buffer Header Area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill Equation 41 on page 2594.

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 131)$$

Equation 41

54.7.4.5 FIFO Message Buffer Header Area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill the following equation:Equation 42 on page 2594.

$$SADR\_MBHF = (i\gamma 8) + FR\_RFSYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 42

2. The message buffer header fields for each FIFO have to be a contiguous area.

54.7.4.6 Message Buffer Data Area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

54.7.4.7 Sync Frame Table Area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to Sync Frame ID and Sync Frame Deviation Tables for the description of the sync frame table area.

54.7.5 Physical Message Buffer Description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer, the message buffer header field and the message buffer data field.

54.7.5.1 Message Buffer Protection and Data Consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

54.7.5.2 Message Buffer Header Field Description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in Message Buffer Header Field. Each message buffer header field consists of two sections: the frame header section and the slot status section.

54.7.5.2.1 Frame Header Description

Frame header content, access, and checks are discussed in this section.

54.7.5.2.1.1 Frame Header Content

The semantic and content of the frame header section depends on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in the figure below. A detailed description is given in Table 54-22.

Table 54-18. Frame Header Structure (Receive Message Buffer and Receive FIFO)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2	0	0	CYCCNT						0	PLDLEN						
0x4	0	0	0	0	0	HDCRC										

The structure of the frame header in the message buffer header field for transmit message buffers is given in the following figure. A detailed description is given in Table 54-23. The checks that will be performed are described in Frame Header Checks.

Table 54-19. Frame Header Structure (Transmit Message Buffer)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID [1]										
0x2			CYCCNT							PLDLEN [2]						
0x4						HDCRC										

= not used

1. checked
2. checked if not static

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in the figure below.

Table 54-20. Frame Header Structure (Transmit Message Buffer for Key Slot)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2			CYCCNT							PLDLEN						
0x4						HDCRC										

= not used

54.7.5.2.1.2 Frame Header Access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in the following table. This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 54-21. Frame Header Write Access Constraints (Transmit Message Buffer)

Field	Static Segment	Dynamic Segment
FID	*POC:config* or MB_DIS	
PPI, PLDLEN, HDCRC	*POC:config* or MB_DIS or	MB_LCK

54.7.5.2.1.3 Frame Header Checks

As shown in Table 54-19 and Table 54-20 not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding Message Buffer Frame ID Registers (FR_MBFIDRn). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag FID_EF in the CHI Error Flag Register (FR_CHIERFR). The value of the FID field will be ignored and replaced by the value provided in the Message Buffer Frame ID Registers (FR_MBFIDRn).

For transmit message buffers assigned to the *static* segment, the PLDLEN value must be equal to the value of the payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). If this is not fulfilled, the static payload length error flag SPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). If this is not fulfilled, the dynamic payload length error flag DPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 54-22. Frame Header Field Descriptions (Receive Message Buffer and Receive FFO)

Field	Description
R	**Reserved Bit** — This is the value of the *Reserved bit* of the received frame stored in the message buffer
PPI	**Payload Preamble Indicator** — This is the value of the *Payload Preamble Indicator* of the received frame stored in the message buffer.
NUF	**Null Frame Indicator** — This is the value of the *Null Frame Indicator* of the received frame stored in the message buffer.
SYF	**Sync Frame Indicator** — This is the value of the *Sync Frame Indicator* of the received frame stored in the message buffer.
SUF	**Startup Frame Indicator** — This is the value of the *Startup Frame Indicator* of the received frame stored in the message buffer.
FID	**Frame ID** — This is the value of the *Frame ID* field of the received frame stored in the message buffer.
CYCCNT	**Cycle Count** — This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	**Payload Length** — This is the value of the *Payload Length* field of the received frame stored in the message buffer.
HDCRC	**Header CRC** — This is the value of the *Header CRC* field of the received frame stored in the message buffer.

Table 54-23. Frame Header Field Descriptions (Transmit Message Buffer)

Field	Description
R	**Reserved Bit** — This bit is not used, the value of the *Reserved bit* is generated internally according to *FlexRay Communications System Protocol Specification, Version*2.1 Rev A.
PPI	**Payload Preamble Indicator** — This bit provides the value of the *Payload Preamble Indicator* for the frame transmitted from the message buffer.
NUF	**Null Frame Indicator** — This bit is not used, the value of the *Null Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SYF	**Sync Frame Indicator** — This bit is not used, the value of the *Sync Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SUF	**Startup Frame Indicator** — This bit is not used, the value of the *Startup Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
FID	**Frame ID** — This field is checked as described in Frame Header Checks.
CYCCNT	**Cycle Count** — This field is not used, the value of the transmitted *Cycle Count* field is taken from the internal communication cycle counter.
PLDLEN	**Payload Length** — This field is checked and used as described in Frame Header Checks.

Table continues on the next page...

Table 54-23. Frame Header Field Descriptions (Transmit Message Buffer) (continued)

Field	Description
HDCRC	**Header CRC** — This field provides the value of the *Header CRC* field for the frame transmitted from the message buffer.

54.7.5.2.2 Slot Status Description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depends on the message buffer type.

54.7.5.2.2.1 Receive Message Buffer and Receive FIFO Slot Status Description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given in the next table.

Table 54-24. Receive Message Buffer Slot Status Content

Receive Message Buffer Type	Slot Status Content
Individual Receive Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 54-25
Individual Receive Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 54-26
Individual Receive Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 54-27
Receive FIFO Channel A Message Buffer	see Table 54-26
Receive FIFO Channel B Message Buffer	see Table 54-27

The meaning of the bits in the slot status structure is explained in the Receive Message Buffer Slot Status Field Description table below.

Table 54-25. Receive Message Buffer Slot Status Structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	CH	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 54-26. Receive Message Buffer Slot Status Structure (ChA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table continues on the next page...

Table 54-26. Receive Message Buffer Slot Status Structure (ChA) (continued)

R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 54-27. Receive Message Buffer Slot Status Structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	1	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 54-28. Receive Message Buffer Slot Status Field Description

Field	Description
Common Message Buffer Status Bits	
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
CH	**Channel first valid received** — This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the *first valid* frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 first valid frame received on channel A, or no valid frame received at all

Table continues on the next page...

Table 54-28. Receive Message Buffer Slot Status Field Description (continued)

Field	Description
	1 first valid frame received on channel B
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1

54.7.5.2.2.2 Transmit Message Buffer Slot Status Description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given in the table below.

Table 54-29. Transmit Message Buffer Slot Status Content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 54-30

Table continues on the next page...

Table 54-29. Transmit Message Buffer Slot Status Content (continued)

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 54-31
Individual Transmit Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 54-32

The meaning of the bits in the slot status structure is described in the Transmit Message Buffer Slot Status Structure Field Descriptions table.

Table 54-30. Transmit Message Buffer Slot Status Structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 54-31. Transmit Message Buffer Slot Status Structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 54-32. Transmit Message Buffer Slot Status Structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 54-33. Transmit Message Buffer Slot Status Structure Field Descriptions

Field	Description
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1

Table continues on the next page...

Table 54-33. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCB	**Transmission Conflict on Channel B** — protocol related variable: *vSS!TxConflict* channel B 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0

Table continues on the next page...

Table 54-33. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
	1 *vSS!BViolation* = 1
TCA	**Transmission Conflict on Channel A** — protocol related variable: *vSS!TxConflict* channel A 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

54.7.5.3 Message Buffer Data Field Description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in the table below. The structure of the message buffer data field is given in the next figure.

Table 54-34. Message Buffer Data Field Minimum Length

physical message buffer assigned to	minimum length defined by
Individual Message Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Receive Shadow Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Individual Message Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive Shadow Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note

The CC will not access any locations outside the message buffer data field boundaries given in the table above.

Table 54-35. Message Buffer Data Field Structure

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	DATA0 / MID0 / NMV0								DATA1 / MID1 / NMV1							
0x2	DATA2 / NMV2								DATA3 / NMV3							
…	…								…							
0xN-2	DATA N-2								DATA N-1							

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

54.7.5.3.1 Message Buffer Data Field Read Access

For transmit message buffers, the CC will not modify the content of the Message Buffer Data Field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the Message Buffer Index Registers (FR_MBIDXRn). While the message buffer is locked, the CC will not update the Message Buffer Data Field.

For receive FIFOs, the application can read the message buffer indicated by the Receive FIFO A Read Index Register (FR_RFARIR) or the Receive FIFO B Read Index Register (FR_RFBRIR) when the related fill levels in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) indicate an non-empty FIFO.

54.7.5.3.2 Message Buffer Data Field Write Access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in the Frame Data Write Access Constraints table below.

Table 54-36. Frame Data Write Access Constraints

Field	CC/MB State
DATA, MID, NMV	*POC:config* or MB_DIS or MB_LCK

Table 54-37. Frame Data Field Descriptions

Field	Description
DATA 0, DATA 1, … DATA N-1	**Message Data** — Provides the message data received or to be transmitted.For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	**Message Identifier** — If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.

Table continues on the next page...

Table 54-37. Frame Data Field Descriptions (continued)

Field	Description
NMV 0, NMV 1, … NMV 11	**Network Management Vector** — If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. **Note:** The MID and NMV bytes replace the corresponding DATA bytes.

54.7.6 Individual Message Buffer Functional Description

The CC provides these basic types of individual message buffers:

1. Transmit Message Buffers
2. Receive Message Buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in Individual Message Buffer Configuration Data.

54.7.6.1 Individual Message Buffer Configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

54.7.6.1.1 Common Configuration Data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. These data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the Message Buffer Data Size Register (FR_MBDSR).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the Receive Shadow Buffer Index Register (FR_RSBIR).

54.7.6.1.2 Specific Configuration Data

The second part of the message buffer configuration data is specific for each message buffer.

These data can be changed only when either

- *the protocol is in the POC:config state or*
- *the message buffer is disabled, i.e. FR_MBCCSRn[EDS] = 0*

The individual message buffer type is defined by the MTD and MBT bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) as given in the following table.

Table 54-38. Individual Message Buffer Types

FR_MBCCSRn		Individual Message Buffer Description
MTD	MBT	
0	0	Receive Message Buffer
0	1	Reserved
1	0	Transmit Message Buffer
1	1	Reserved

The message buffer specific configuration data are

1. MTD bits in Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)
2. all fields and bits in Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)

3. all fields and bits in Message Buffer Frame ID Registers (FR_MBFIDRn)
4. all fields and bits in Message Buffer Index Registers (FR_MBIDXRn)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions in the Transmit Message Buffers section and Receive Message Buffers section.

54.7.6.2 Transmit Message Buffers

The section provides a detailed description of the functionality of single buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number n is configured to be a transmit message buffer by the following settings:

- FR__ MBCCSRn[MBT] = 0(singlebufferedmessagebuffer)
- FR_ MBCCSRn[MDT] = 1(transmitmessagebuffer)

54.7.6.2.1 Access Regions

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented, which is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in the figure below. A description of the regions is given in Table 54-40. If an region is active as indicated in Table 54-41, the access scheme given for that region applies to the message buffer.

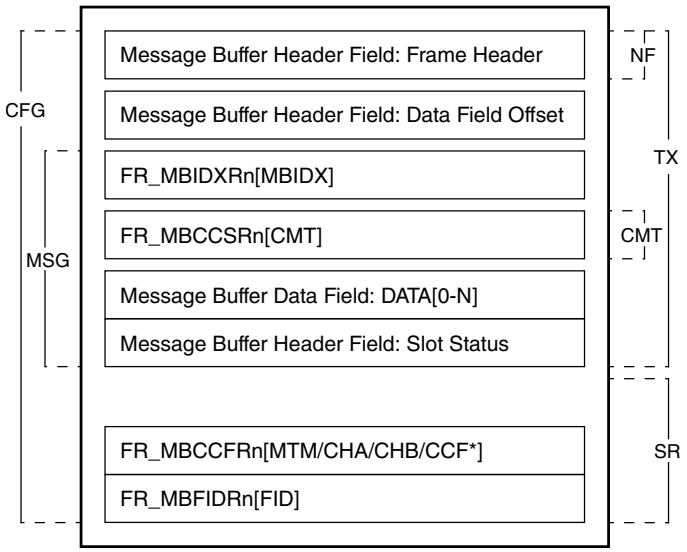


Figure 54-8. Transmit Message Buffer Access Regions

Table 54-39. Transmit Message Buffer Access Regions Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration
MSG	read/write	-	Message Data and Slot Status Access
NF	-	read-only	Message Header Access for Null Frame Transmission
TX	-	read/write	Message Transmission and Slot Status Update
CM	-	read-only	Message Buffer Validation
SR	-	read-only	Message Buffer Search

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT], and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in the next figure. A description of the states is given in Table 54-40, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

54.7.6.2.2 Message Buffer States

This section describes the transmit message buffer states and provides a state diagram.

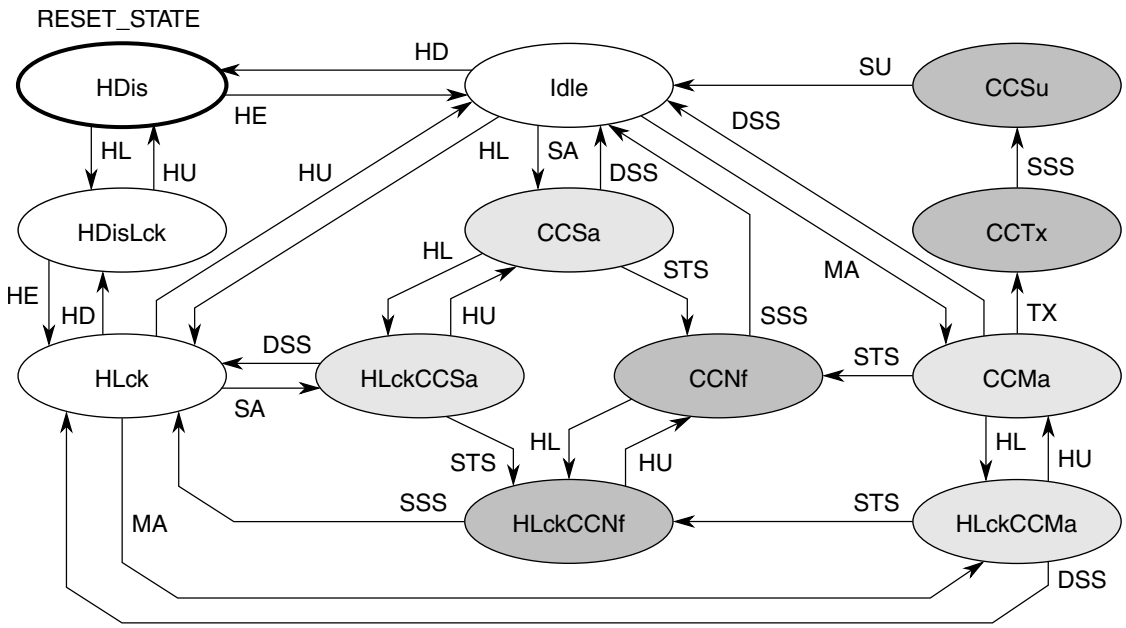


Figure 54-9. Transmit Message Buffer States

Table 54-40. Transmit Message Buffer State Description (Sheet 1 of 2)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	CM,SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	SR	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCSa	1	0	–	–	**Slot Assigned** - Message buffer assigned to next static slot.Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	–	**Locked and Slot Assigned** - Applications access to data, control, and status.Message buffer assigned to next static slot
CCNf	1	0	–	NF	**Null Frame Transmission** Header is used for null frame transmission.

Table continues on the next page...

Table 54-40. Transmit Message Buffer State Description (Sheet 1 of 2) (continued)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
HLckCCNf	1	1	MSG	NF	**Locked and Null Frame Transmission** - Applications access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	–	CM	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
HLckCCMa	1	1	MSG	–	**Locked and Message Available** - Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	–	TX	**Message Transmission** - Message buffer data transmit. Payload data from buffer transmitted
CCSu	1	0	–	TX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

54.7.6.2.3 Message Buffer Transitions

Application transitions, module transitions, and transition priorities are discussed in this section.

54.7.6.2.3.1 Application Transitions

The application transitions can be triggered by the application using the commands described in the table below. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

54.7.6.2.3.1.1 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCSa, HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

54.7.6.2.3.1.2 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 54-41. Transmit Message Buffer Application Transitions

Transition	Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]:= 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

54.7.6.2.3.2 Module Transitions

The module transitions that can be triggered by the CC are described in the following table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 54-42. Transmit Message Buffer Module Transitions

Transition	Condition	Description
SA	slot match andstatic slot	**Slot Assigned** - Message buffer is assigned to next static slot.
MA	slot match andCycleCounter match	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
TX	slot start and FR_MBCCSRn[CMT] = 1	**Transmission Slot Start** - Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	status updated	**Status Updated** - Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	static slot start	**Static Slot Start** - Start of static slot.
DSS	dynamic slot start orsymbol window start orNIT start	**Dynamic Slot or Segment Start.** - Start of dynamic slot or symbol window or NIT.
SSS	slot start orsymbol window start orNIT start	**Slot or Segment Start** - Start of static slot or dynamic slot or symbol window or NIT.

54.7.6.2.3.3 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of the next table, the module transitions have a higher priority than the application transitions. For all states except the CCMa state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and subsequently the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions is given in the second part of the table below.

Table 54-43. Transmit Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle, HLck	SA > HD	Slot Assigned > Message Buffer Disable
	MA > HD	Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS	Transmission Slot Start > Static Slot Start
	TX > DSS	Transmission Slot Start > Dynamic Slot Start

54.7.6.2.4 Transmit Message Setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit CMT in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). The physical access to the message buffer data field is described in Individual Message Buffers.

As indicated by Table 54-33, the application shall write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in Table 54-33. The state change is indicated through the FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa the FR_MBCCSRn[DVAL] flag is negated.

54.7.6.2.5 Message Transmission

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. the message buffer is in the message available state CCMa
2. the message data are still valid, i.e. FR_MBCCSRn[CMT] = 1

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in the "Message Transmission Timing" figure below. In this example, the message buffer with message buffer number n is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid, i.e. FR_MBCCSRn[CMT] = 1.

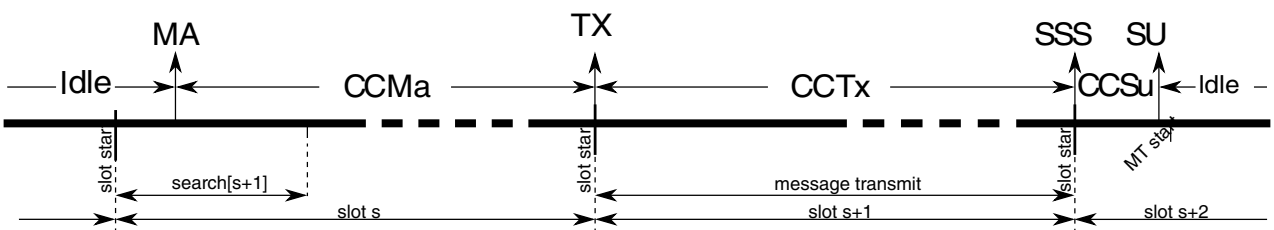


Figure 54-10. Message Transmission Timing

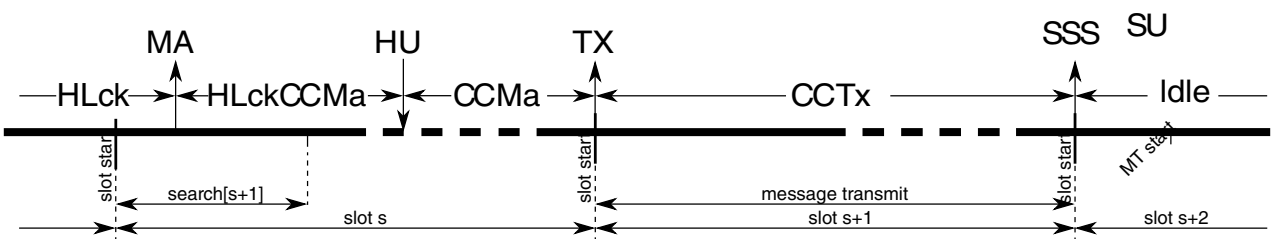


Figure 54-11. Message Transmission from HLck state with unlock

The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)

3. the value of the PLDLEN field in the message buffer header field, as described in Frame Header Description

If a message buffer is assigned to message buffer segment 1, and PLDLEN > MBSEG1DS, then 2 × MBSEG1DS bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If PLDLEN <= MBSEG1DS, the CC reads and transfers 2 ×PLDLEN bytes. The same holds for segment 2 and MBSEG2DS.

54.7.6.2.6 Null Frame Transmission

A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the FR_MBFIDRn[FID] set to S and FR_MBCCFRn[CHA] set to 1. A Null Frame is transmitted in the static slot S on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with FR_MBFIDRn[FID] = s and FR_MBCCFRn[CHA] = 1 are either not committed, i.e FR_MBCCSRn[CMT] = 0, or locked by the application, i.e. FR_MBCCSRn[LCKS] = 1, or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in the figure below.

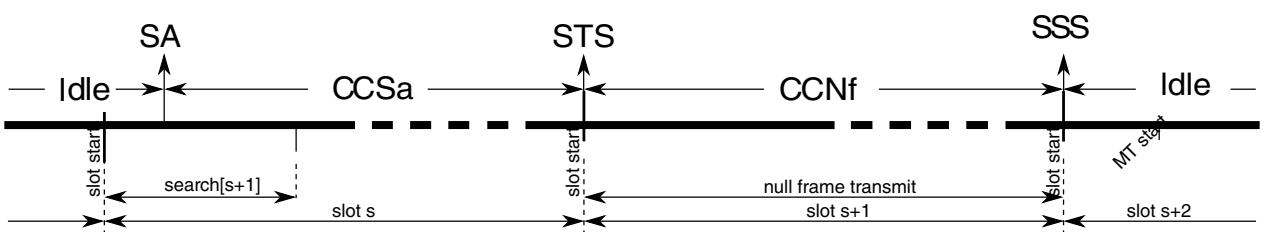


Figure 54-12. Null Frame Transmission from Idle state

A message buffer timing and state change diagram for null frame transmission from HLck state is given in the following figure.

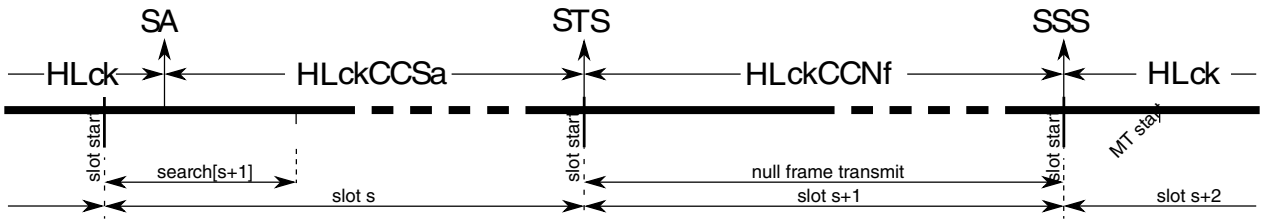


Figure 54-13. Null Frame Transmission from HLck state

If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the next figure.

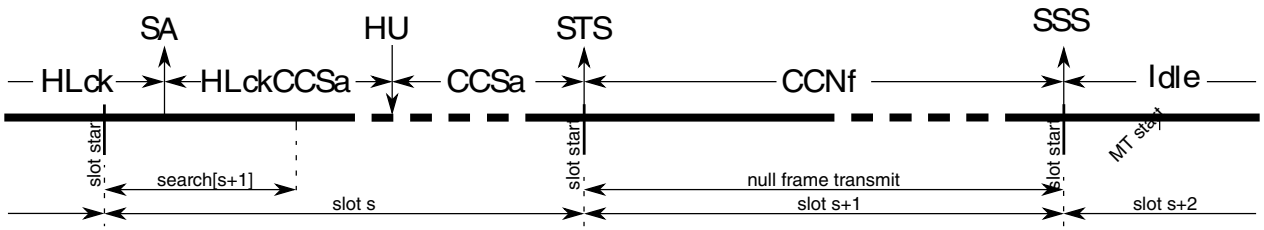


Figure 54-14. Null Frame Transmission from HLck state with unlock

Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the figure below.

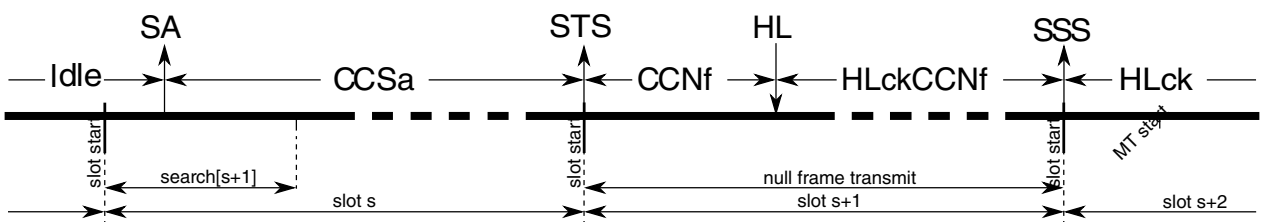


Figure 54-15. Null Frame Transmission from Idle state with locking

54.7.6.2.7 Message Buffer Status Update

After the end of each slot, the PE generates the slot status vector. Depending on the this status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

54.7.6.2.7.1 Message Buffer Status Update after Complete Message Transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer were send to FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSRn[MBIF] to indicate the successful message transmission.

Depending on the transmission mode flag FR_MBCCFRn[MTM], the CC changes the commit flag FR_MBCCSRn[CMT] and the valid flag FR_MBCCSRn[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in the *event transmission mode*. In this case, each committed message is transmitted only once. The commit flag FR_MBCCSRn[CMT] is cleared with the SU transition. If the FR_MBCCFRn[MTM] flag is asserted, the message buffer is in the *state transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSRn[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSRn[DVAL] to indicate that the message will be transmitted again.

54.7.6.2.7.2 Message Buffer Status Update after Incomplete Message Transmission

The term incomplete message transmission refers to the fact that not all payload data that should be transmitted were send to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than *pLatestTx*.
2. The transmission slot did not exist in the dynamic segment at all.

In any of these two aforementioned conditions occur, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those errors occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the Protocol Interrupt Flag Register 1 (FR_PIFR1).

54.7.6.2.7.3 Message Buffer Status Update after Null Frame Transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

54.7.6.3 Receive Message Buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used it is required to configure the related receive shadow buffer as described in Receive Shadow Buffers.

A receive message buffer is used to receive a message from the FlexRay Bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application.

1. message data received
2. information about the reception process
3. status information about the slot in which the message was received

A individual message buffer with message buffer number *n* is configured as a receive message buffer by the following configuration settings

$$FR\_MBCCSRn[MTD] = 0 (receive message buffer)$$

Equation 43

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented that is used to control the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in the figure below. A description of the regions is given in the following table. If a region is active as indicated in the "Receive Message Buffer States and Access (Sheet 2 of 2)" table below, the access scheme given for that region applies to the message buffer.

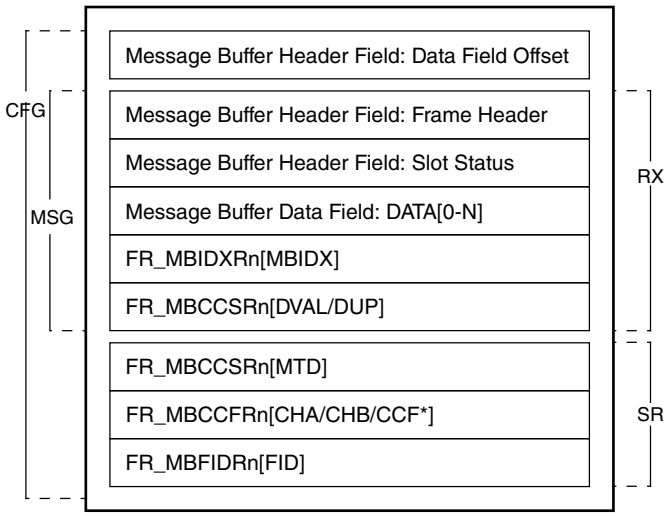


Figure 54-16. Receive Message Buffer Access Regions

Table 54-44. Receive Message Buffer Access Region Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration, Message Data and Status Access
MSG	read/write	-	Message Data, Header, and Status Access
RX	-	write-only	Message Reception and Status Update
SR	-	read-only	Message Buffer Search Data

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT] and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in the figure below. A description of the message buffer states is given in Table 54-40, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

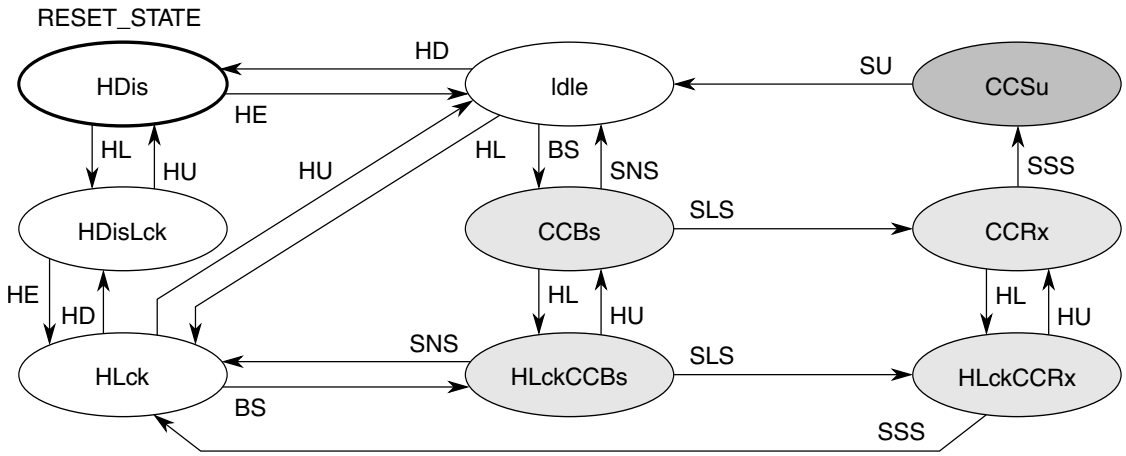


Figure 54-17. Receive Message Buffer States

Table 54-45. Receive Message Buffer States and Access (Sheet 2 of 2)

State	FR_MBCCSRn		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	–	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCBs	1	0	–	–	**Buffer Subscribed** - Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	–	**Locked and Buffer Subscribed** - Applications access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	–	–	**Message Receive** - Message data received into related shadow buffer.
HLckCCRx	1	1	MSG	–	**Locked and Message Receive** - Applications access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	–	RX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

54.7.6.3.1 Message Buffer Transitions

Application transitions, message buffer enable and disable, message buffer lock and unlock, module transitions, and transition priorities are discussed in this section.

54.7.6.3.1.1 Application Transitions

The application transitions that can be triggered by the application using the commands described in Table 54-41. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

54.7.6.3.1.2 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCBs, HLckCCBs, or CCRx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

54.7.6.3.1.3 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL while the message buffer is in the state CCRx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 54-46. Receive Message Buffer Application Transitions

Transition	Host Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]: = 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

54.7.6.3.1.4 Module Transitions

The module transitions that can be triggered by the CC are described in the next table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 54-47. Receive Message Buffer Module Transitions

Transition	Condition	Description
BS	slot match and CycleCounter match	**Buffer Subscribed** - The message buffer filter matches next slot and cycle.
SLS	slot start	**Slot Start** - Start of either Static Slot or Dynamic Slot.
SNS	symbol window start or NIT start	**Symbol Window or NIT Start** - Start of either Symbol Window or NIT.
SSS	slot start or symbol window start or NIT start	**Slot or Segment Start** - Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	status updated	**Status Updated** - Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

54.7.6.3.1.5 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the table below, the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU and can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the buffer subscribed state CCBs and the module triggers the slot start transition SLS at the same time as the application locks the message buffer by the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state HLckCCRx.

Table 54-48. Receive Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle	BS > HD	Buffer Subscribed > Message Buffer Disable
HLck	BS > HD	Buffer Subscribed > Message Buffer Disable
CCRx	SSS > HL	Slot or Segment Start > Message Buffer Lock

54.7.6.3.2 Message Reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to the either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffers assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot the module trigger the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data are written into the shadow buffers. For details on receive shadow buffers, see Receive Shadow Buffers Concept. The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

54.7.6.3.3 Message Buffer Update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer was in the locked state HLckCCRx, no update will be performed. The received data are lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the CHI Error Flag Register (FR_CHIERFR).

If a message buffer was in the CCRx state it is now in the CCSu state. After the evaluation of the slot status provided by the PE the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to. This is described in Table 54-51.

Table 54-49. Receive Message Buffer Update (Continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
1	1	**Valid non-null frame received.** - Message Buffer Data Field updated. - Frame Header Field updated. - Slot Status Field updated. - DUP:= 1 - DVAL:= 1 - MBIF:= 1
1	0	**Valid null frame received.**

Table continues on the next page...

Table 54-49. Receive Message Buffer Update (Continued) (continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
		- Message Buffer Data Field *not* updated. - Frame Header Field *not* updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed - MBIF:= 1
0	x	**No valid frame received.** - Message Buffer Data Field not updated. - Frame Header Field not updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed. - MBIF:= 1, if the slot was not an empty dynamic slot. **Note:** An empty dynamic slot is indicated by the following frame and slot status bit values: *vSS!ValidFrame* = 0 and *vSS!SyntaxError* = 0 and *vSS! ContentError* = 0 and *vSS!BViolation* = 0.

Note

If the number of the last slot in the current communication cycle on a given channel is n, then all receive message buffers assigned to this channel with FR_MBFIDRn[FID] > *n* will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in the following figure.

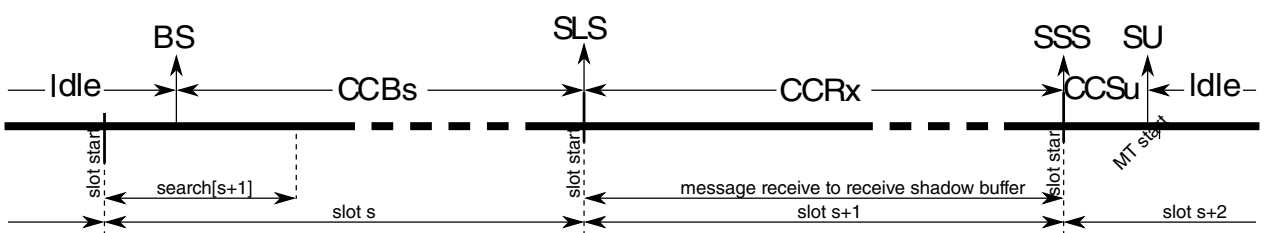


Figure 54-18. Message Reception Timing

The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following two items:

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. the number of bytes received over the FlexRay bus

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than 2 ×FR_MBDSR.MBSEG1DS, the CC writes only 2 × FR_MBDSR.MBSEG1DS bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than 2 × FR_MBDSR.MBSEG1DS, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with FR_MBDSR.MBSEG2DS.

54.7.6.3.4 Received Message Access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in Individual Message Buffers.

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

54.7.6.3.5 Receive Shadow Buffers Concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in Receive Shadow Buffers.

The receive shadow buffers temporarily store the received frame header and message data. After the slot boundary the slot status information is generated. If the slot status information indicates the reception of the valid non-null frame (see Table 54-49), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the Message Buffer Index Registers (FR_MBIDXRn) with the content of the corresponding internal shadow buffer index register. In all other cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the Message Buffer Index Registers (FR_MBIDXRn). Instead, the index of the message buffer header field must be fetched from the Message Buffer Index Registers (FR_MBIDXRn).

54.7.7 Individual Message Buffer Search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel if a slot s in a communication cycle c is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm which is invoked at the following protocol related events:

1. NIT start
2. slot start in the static segment
3. minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot *n* searches for message buffers assigned or subscribed to slot *n+1*.

In general, the message buffer search for the next slot *n* considers only message buffers which are

1. enabled, i.e. FR_MBCCSRn[EDS] = 1, and
2. matches the next slot *n*, i.e. FR_MBFIDRn[FID] = *n*, and

On top of that, for the static segment only those message buffers are considered, that match the condition of at least one row of the "Message Buffer Search Priority (static segment)" table shown below. For the dynamic segment only those message buffers are considered, that match the condition of at least one row of the next "Message Buffer Search Priority (dynamic segment)" table. These message buffers are called *matching* message buffers.

For each enabled channel the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers the message buffers with highest priority according to the "Message Buffer Search Priority (static segment)" table for the static segment and according to the "Message Buffer Search Priority (dynamic segment)" table for the dynamic segment are selected.

Table 54-50. Message Buffer Search Priority (static segment)

Priority	MTD	LCKS	CMT	CCFM-[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	1	-	0	1	transmit buffer, matches cycle count, not committed	SA
	1	1	-	1	transmit buffer, matches cycle count, locked	SA
2	1	-	-	-	transmit buffer	SA
3	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 4	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.
2. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

Table 54-51. Message Buffer Search Priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM-[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 2	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.
2. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

If there are multiple message buffer with highest priority, the message buffer with the lowest message buffer number is selected. All message buffer which have the highest priority must have a consistent channel assignment as specified in Message Buffer Channel Assignment Consistency.

Depending on the message buffer channel assignment the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in Individual Message Buffers.

54.7.7.1 Message Buffer Cycle Counter Filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn). This filter determines a set of communication cycles in which the message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled, i.e. CCFE = 0, this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following conditions evaluates to true:

$$MBCCFRn[CCFE] = 0$$

$$CYCCNT \,\&\, MBCCFRn[CCFMSK] = MBCCFRn[CCFVAL] \,\&\, MBCCFRn[CCFMSK]$$

54.7.7.2 Message Buffer Channel Assignment Consistency

The message buffer channel assignment given by the CHA and CHB bits in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn) defines the channels on which the message buffer will receive or transmit. The message buffer with number *n* transmits or receives on channel A if FR_MBCCFRn[CHA] = 1 and transmits or receives on channel B if FR_MBCCFRn[CHB] = 1.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they must be either assigned to one channel only, or must be assigned to *both* channels. The behavior of the message buffer search is not defined, if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in the figure below.

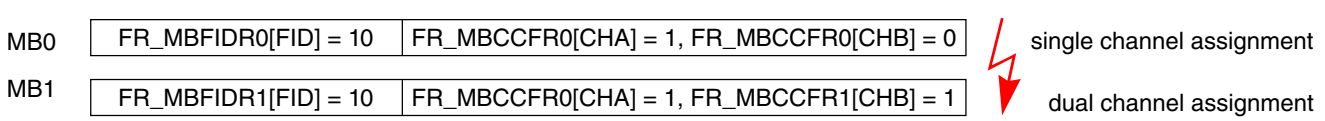


Figure 54-19. Inconsistent Channel Assignment

54.7.7.3 Node Related Slot Multiplexing

The term *Node Related Slot Multiplexing* applies to the dynamic segment only and refers to the functionality if there are transmit as well as receive message buffers are configured for the same slot.

According to Table 54-51 the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

54.7.7.4 Message Buffer Search Error

There are two kinds of errors which may occur during message buffer search[1].

54.7.7.4.1 Message Buffer Search Start while Running

If the message buffer search is running in slot *n-1* and the next message buffer search start event appears due to the start of slot *n*, the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR). As a result of this stop, no individual message buffer is identified for transmission or reception in slot *n*. Additionally, the search engine will not be started in slot *n*, and consequently no individual message buffer is identified for transmission or reception in slot *n+1*.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency see Number of Usable Message Buffers.

54.7.7.4.2 Illegal Message Buffer Index Found

If the message buffer search has finished the message buffer search in slot *n-1*, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR) is set and no individual message buffer is identified for transmission or reception in slot

1. The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to an search error, the received frame is considered for FIFO reception.

n. The legal message buffer index values for the individual and receive shadow buffers are specified in the Receive Shadow Buffer Index Register section (FR_RSBIR) and the Message Buffer Index Registers sections (FR_MBIDXRn).

54.7.8 Individual Message Buffer Reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on Specific Configuration Data. The common configuration data given in the section on Specific Configuration Data can not be reconfigured when the protocol is out of the *POC:config* state.

54.7.8.1 Reconfiguration Schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

54.7.8.1.1 Basic Type Not Changed (RC1)

A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSRn[MTD] are not changed. This type of reconfiguration is denoted by RC1 in Figure 54-20. Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

54.7.8.1.2 Buffer Type Not Changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in the figure below. It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

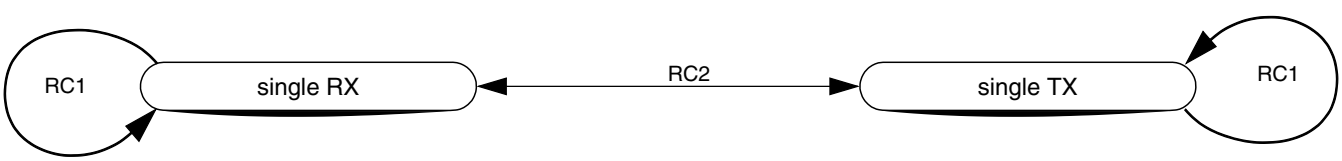


Figure 54-20. Message Buffer Reconfiguration Scheme

54.7.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

54.7.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. One FIFO is assigned to channel A, the other FIFO is assigned to channel B. Both FIFOs work completely independent from each other.

The message buffer structure of each FIFO is described in Receive FIFO. The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- *The FIFO system memory base address*
- *The index of the first FIFO entry given by Receive FIFO Start Index Register (FR_RFSIR)*
- *The data field offset of the data field belonging to the first FIFO entry given by Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *The number of FIFO entries and the length of each FIFO entry as given by Receive FIFO Depth and Size Register (FR_RFDSR)*

54.7.9.2 FIFO Configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO address mode bit FAM in the Module Configuration Register (FR_MCR).

54.7.9.2.1 Single System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is System Memory Base Address Register (FR_SYMBADR).

54.7.9.2.2 Dual System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is Receive FIFO System Memory Base Address Register (FR_RFSYMBADR).

The FIFO control and configuration data are given in Receive FIFO Control and Configuration Data. The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details see FlexRay Memory Area Layout.

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

- Configure the FIFO update and address modes in Module Configuration Register (FR_MCR)
- Configure the FIFO system memory base address
- Configure the Receive FIFO Start Index Register (FR_RFSIR) with the first message buffer header index that belongs to the FIFO
- Configure the Receive FIFO Start Data Offset Register (FR_RFSDOR) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO entry size
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO depth
- Configure the FIFO Filters

54.7.9.3 FIFO Periodic Timer

The FIFO periodic timer is used to generate an FIFO almost-full interrupt at certain point in time, if the almost-full watermark is not reached, but the FIFO is not empty. This can be used to prevent frames from get stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the Receive FIFO Periodic Timer Register (FR_RFPTR). If the periodic timer duration FR_RFPTR[PTD] is configured to 0x0000, the periodic timer is continuously expired. If the periodic timer duration FR_RFPTR[PTD] is configured to 0x3FFF, the periodic timer never expires. If the periodic timer is configured to a value *ptd*, greater than 0x0000 and smaller 0x3FFF, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

54.7.9.4 FIFO Reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO, if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case the FIFO filter path shown in Figure 54-21 is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the fifo fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is incremented. If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

54.7.9.5 FIFO Almost-Full Interrupt Generation

If the fifo fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM, i.e. FLA>WM_A (FLB>WM_B), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

If the periodic timer expires, and FIFOA (FIFOB) is not empty, i.e. FLA>0 (FLB>0), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

54.7.9.6 FIFO Overflow Error Generation

If the FIFOA (FIFOB) is full, i.e. FLA=$FIFO\_DEPTH_A$ (FLB=$FIFO\_DEPTH_B$) and the conditions for a FIFO reception as described in FIFO Reception are fulfilled, then the fifo overflow error flag FR_CHIERFR[FOVA_EF] (FR_CHIERFR[FOVB_EF]) is asserted.

54.7.9.7 FIFO Message Access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is greater than 0. The Receive FIFO A Read Index Register (FR_RFARIR) and the

(Receive FIFO B Read Index Register (FR_RFBRIR)) point to a message buffer with valid content and the oldest frames stored in the FIFO.The respective read data field offsets can be calculated according to Equation 1-125.

If the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is 0, than the FIFOA (FIFOB) contains no valid messages and the corresponding read index register Receive FIFO A Read Index Register (FR_RFARIR) or (Receive FIFO B Read Index Register (FR_RFBRIR) point to a message buffer with invalid content. In this case the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to Equation 1-125. The application can access the message data as described in Receive FIFO. When the application has read the message buffer data and status information, it can update the FIFO as described in FIFO Update.

54.7.9.8 FIFO Update

The application updates the FIFOA (FIFOB) by writing a pop count value *pc* different from 0 to the PCA (PCB) field in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).

As a result of the this operation, the CC removes the oldest *pc* entries from FIFOA (FIFOB).

If the specified pop count value *pc* is greater than the current fill level *fl* provided in FLA (FAB) field, then only *fl* entries are removed from the FIFOA (FIFOB), the remaining *fl-pc* requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by the number of removed items. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in Receive FIFO Start Index Register (FR_RFSIR) automatically.

54.7.9.8.1 FIFO Interrupt Flag Update

The FIFO Interrupt Flag Update mode is configured, when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode FIFOA (FIFOB) will be updated by 1 entry, when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by 1, if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

54.7.9.9 FIFO Filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides three sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in the figure below.

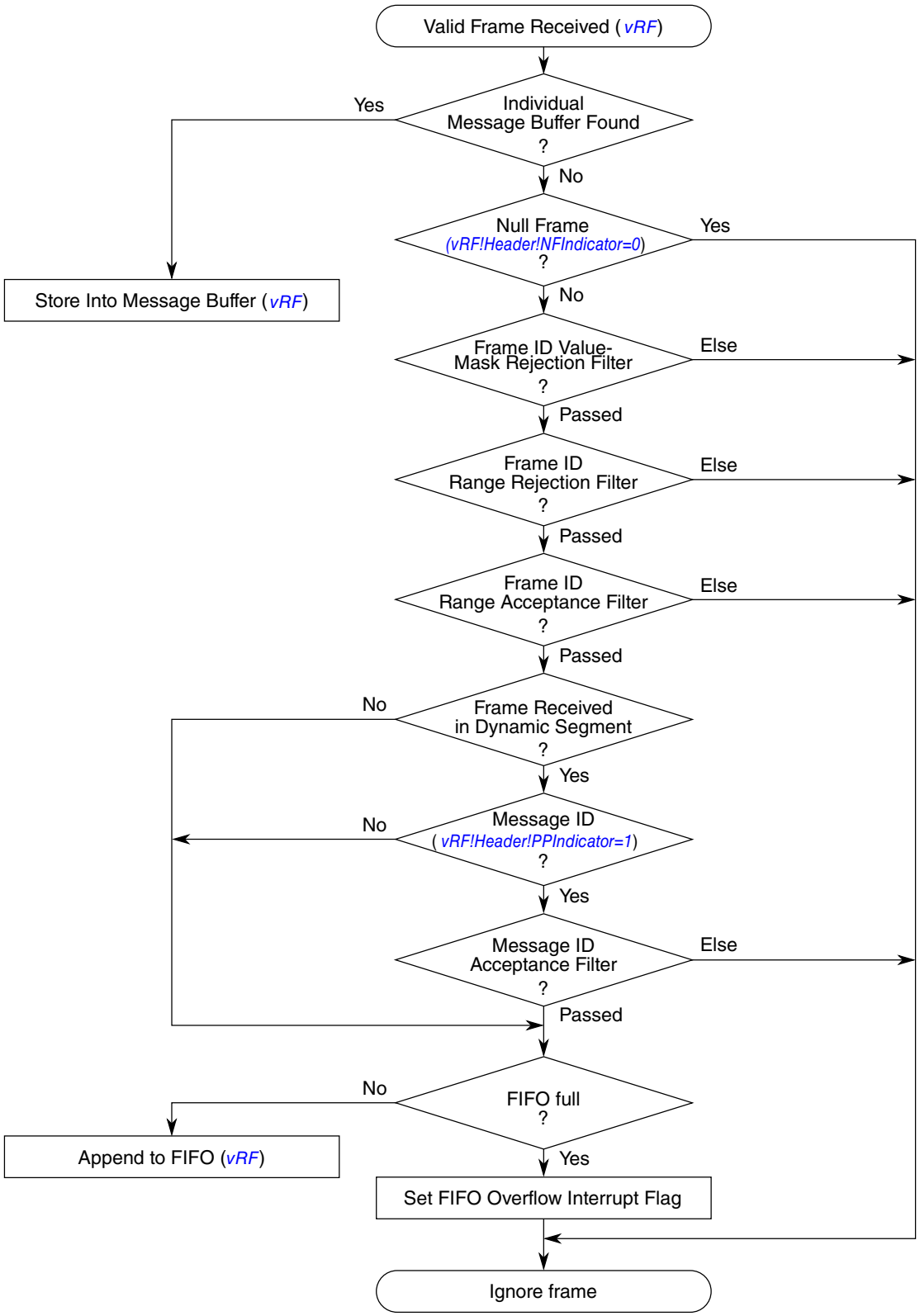


Figure 54-21. Received Frame FIFO Filter Path

A received frame passes the FIFO filtering if it has passed all three type of filter.

54.7.9.9.1 RX FIFO Frame ID Value-Mask Rejection Filter

The frame ID value-mask rejection filter is a value-mask filter and is defined by the fields in the Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR) and the Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter, i.e. is not rejected.

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-Mask Rejection Filter if Equation 44 on page 2637 is fulfilled.

$$FID \,\&\, FR\_RFFIDRFMR[FIDRFMSK] \neq FR\_RFFIDRFVR[FIDRFVAL] \,\&\, FR\_RFFIDRFMR[FIDRFMSK]$$

Equation 44

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to pass all frames by the following settings.

$$FR\_RFFIDRFVR[FIDRFVAL] = 0x000 \text{ and } FR\_RFFIDRFMR[FIDRFMSK] = 0x7FF$$

Equation 45

Using the settings above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to reject all frames by the following settings.

$$FR\_RFFIDRFMR[FIDRFMSK] = 0x000$$

Equation 46

Using these settings, Equation 44 on page 2637 can never be fulfilled (0!= 0) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

54.7.9.9.2 RX FIFO Frame ID Range Rejection Filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range rejection filters if either no rejection filter is enabled, or, for all of the enabled RX FIFO Frame ID Range rejection filters, i.e. FR_RFRFCTR[FiMD] = 1 and FR_RFRFCTR[FiEN] = 1, Equation 47 on page 2638 is fulfilled.

$$FID < FR\_RFRFCFR_{SEL}[SID_{IBD=0}]) or (FR\_RFRFCFR_{SEL}[SID_{IBD=1}] < FID)$$

Equation 47

Consequently, all frames with a frame ID that fulfills Equation 48 on page 2638 for at least one of the enabled rejection filters will be rejected and thus not pass.

$$FR\_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR\_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 48

54.7.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, for at least one of the enabled RX FIFO Frame ID Range acceptance filters, i.e. FR_RFRFCTR[FiMD] = 0 and FR_RFRFCTR[FiEN] = 1, Equation 49 on page 2638 is fulfilled.

$$FR\_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR\_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 49

54.7.9.9.4 RX FIFO Message ID Acceptance Filter

The RX FIFO Message ID Acceptance Filter is a value-mask filter and is defined by the Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR) and the Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR). This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance Filter if Equation 50 on page 2638 is fulfilled.

$$MID \,\&\, FR\_RFMIDAFMR[MIDAFMSK] = FR\_RFMIDAFMR[MIDAFVAL] \,\&\, FR\_RFMIDAFMR[MIDAFMSK]$$

Equation 50

The RX FIFO Message ID Acceptance Filter can be configured to accept all frames by setting

$$FR\_RFMIDAFMR[MIDAFMSK] := 0x000$$

Equation 51

Using these settings, Equation 50 on page 2638 is always fulfilled and all frames will pass.

54.7.10 Channel Device Modes

This section describes the two FlexRay channel device modes that are supported by the CC.

54.7.10.1 Dual Channel Device Mode

In the dual channel device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX, FR_B_TX, and $\overline{\text{FR\_B\_TX\_EN}}$ is connected to the physical bus channel B. The dual channel system is shown in the following figure.

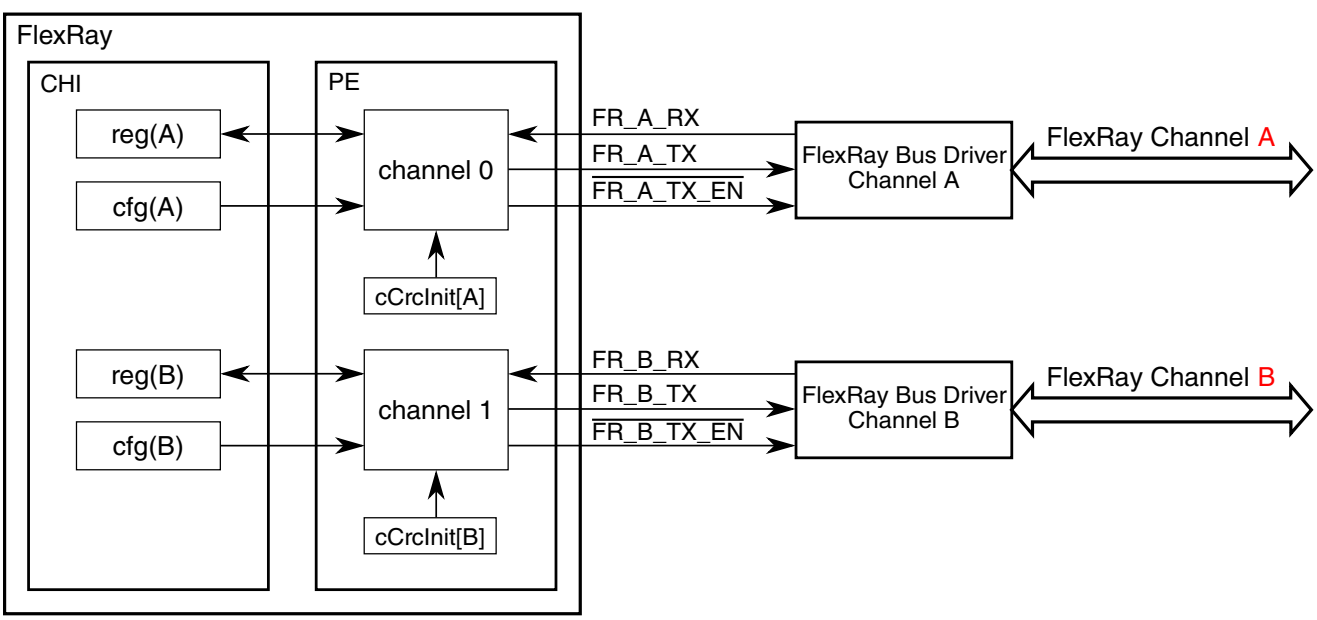


Figure 54-22. Dual Channel Device Mode

54.7.10.2 Single Channel Device Mode

The single channel device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ and can be connected to either the physical bus channel A (shown in the "Single Channel Device Mode (Channel A)" figure below) or the physical bus channel B (shown in the "Single Channel Device Mode (Channel B)" figure below).

If the device is configured as a single channel device by setting FR_MCR[SCM] to 1, only the internal channel A and the FlexRay Port A is used. Depending on the setting of FR_MCR[CHA] and FR_MCR[CHB], the internal channel A behaves either as a FlexRay Channel A or FlexRay Channel B. The bit FR_MCR[CHA] must be set, if the FlexRay Port A is connected to a FlexRay Channel A. The bit FR_MCR[CHB] must be set if the FlexRay Port A is connected to a FlexRay Channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single channel device, the application can access and configure only the registers related to internal channel A.

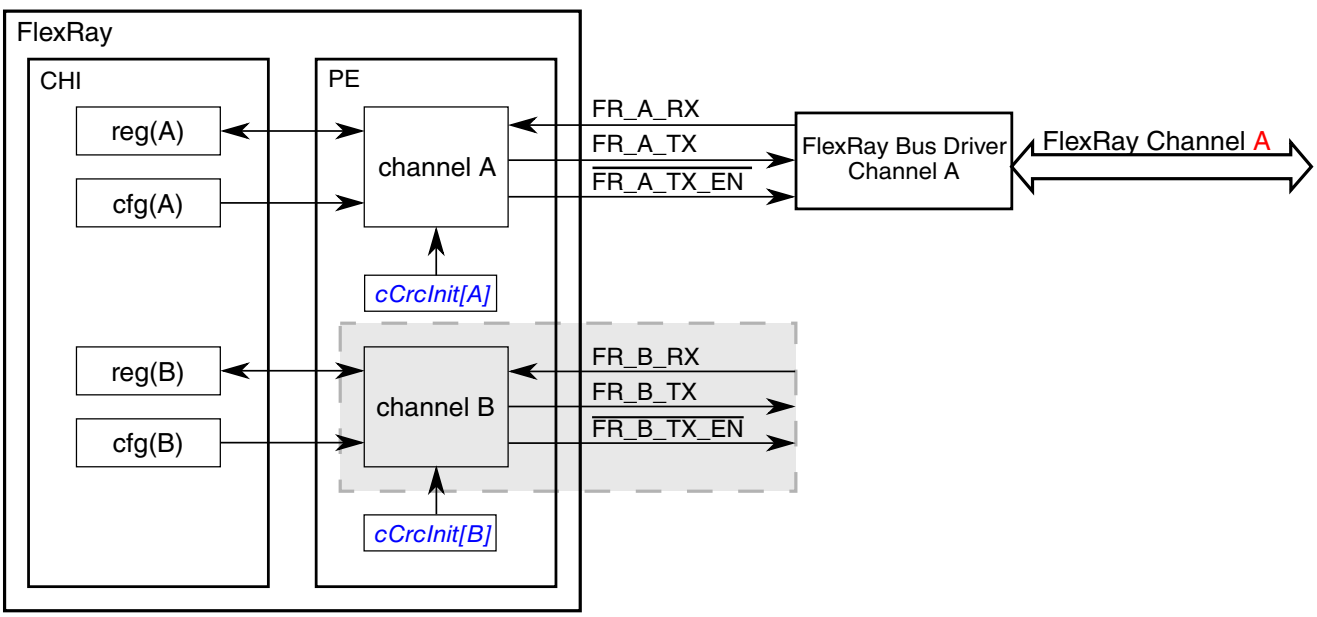


Figure 54-23. Single Channel Device Mode (Channel A)

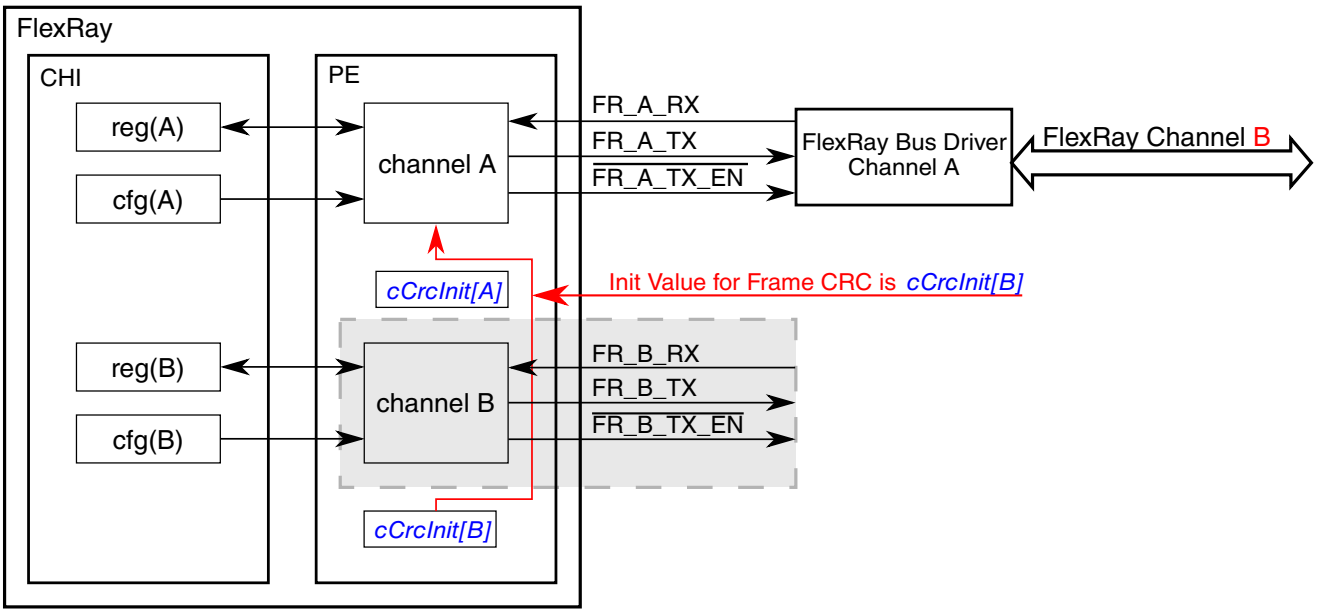


Figure 54-24. Single Channel Device Mode (Channel B)

54.7.11 External Clock Synchronization

The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the Protocol Operation Control Register (FR_POCR). The PE applies the external correction values in the next even-odd cycle pair as shown in the "External Offset Correction Write and Application Timing" figure and the "External Rate Correction Write and Application Timing" figures below.

Note

The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these value were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle 2n+1 shall be affect by the external offset correction, the EOC_AP field must be written to after the start of cycle 2n and before the end of the static segment of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle 2n+1. If the value is not applied in cycle 2n+1, then the value will be applied in the cycle 2n+3. Refer to the following for timing details.

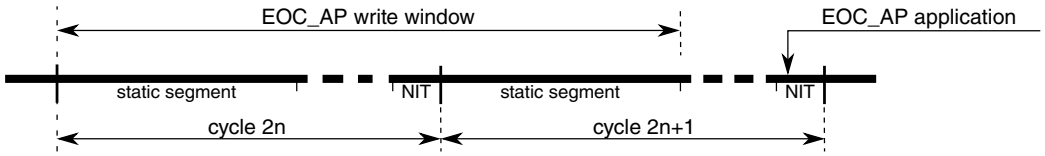


Figure 54-25. External Offset Correction Write and Application Timing

If the rate correction for the cycle pair [2n+2, 2n+3] shall be affect by the external offset correction, the ERC_AP field must be written to after the start of cycle 2n and before the end of the static segment start of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle pair [2n+2, 2n+3]. If the value is not applied for cycle pair [2n+2, 2n+3], then the value will be applied for cycle pair [2n+4, 2n+5]. Refer to the following for details.

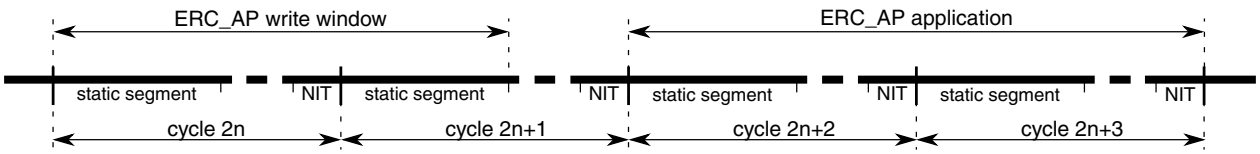


Figure 54-26. External Rate Correction Write and Application Timing

54.7.12 Sync Frame ID and Sync Frame Deviation Tables

The FlexRay protocol requires the provision of a snapshot of the Synchronization Frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note

Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

54.7.12.1 Sync Frame ID Table Content

The Sync Frame ID Table is a snapshot of the protocol related variables *vsSyncIdListA* and *vsSyncIdListB* for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

54.7.12.2 Sync Frame Deviation Table Content

The Sync Frame Deviation Table is a snapshot of the protocol related variable zsDev(id)(oe)(ch)!Value. Each Sync Frame Deviation Table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the Sync Frame ID Table.

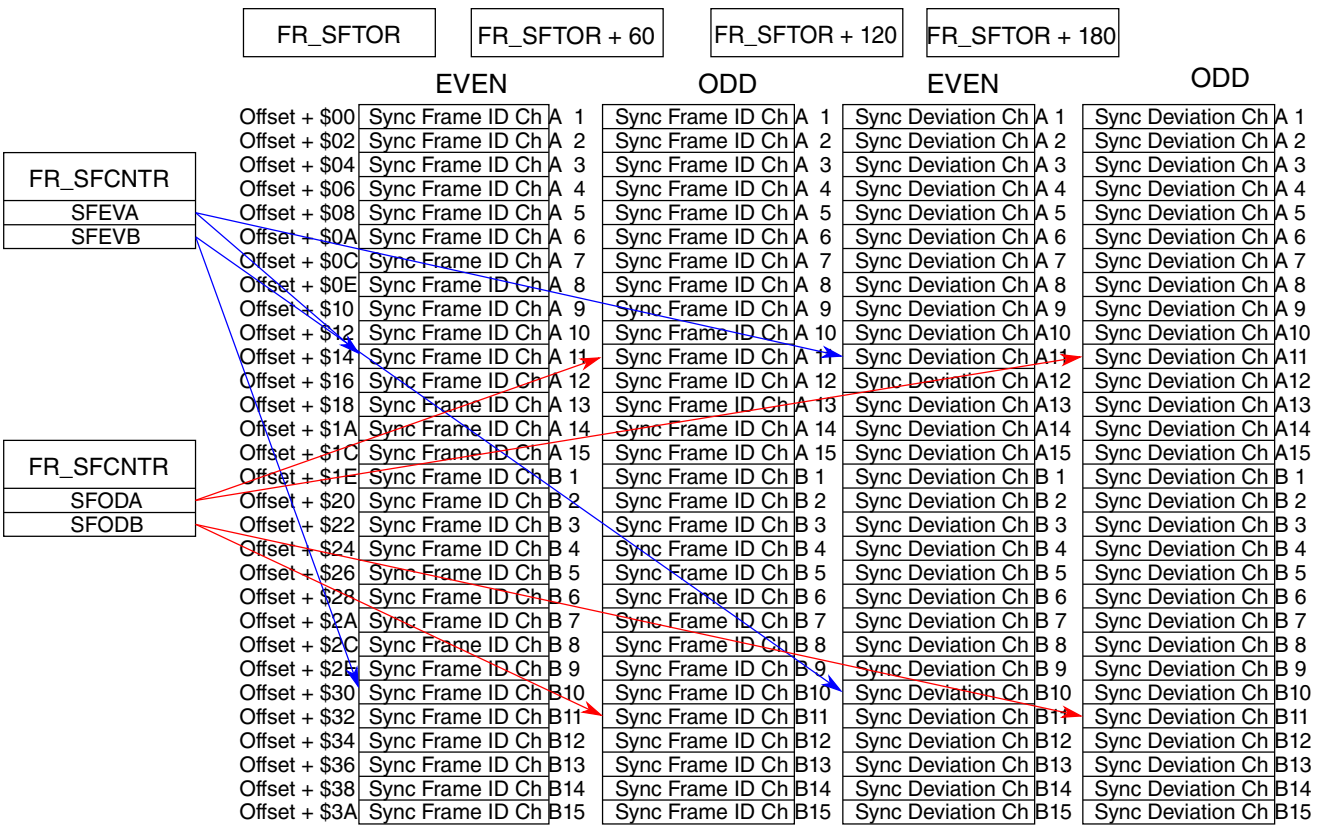


Figure 54-27. Sync Table Memory Layout

54.7.12.3 Sync Frame ID and Sync Frame Deviation Table Setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in Figure 54-27. Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the Sync Frame Table Offset Register (FR_SFTOR).

54.7.12.4 Sync Frame ID and Sync Frame Deviation Table Generation

The application controls the generation process of the Sync Frame ID and Sync Frame Deviation Tables into the FlexRay memory area using the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). A summary of the copy modes is given in the table below.

Table 54-52. Sync Frame Table Generation Modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No Sync Frame Table copy
0	0	1	Sync Frame ID Tables will be copied continuously
0	1	0	Reserved
0	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables will be copied continuously
1	0	0	No Sync Frame Table copy
1	0	1	Sync Frame ID Tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables for next even-odd-cycle pair will be copied

The Sync Frame Table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even cycle related tables after the start of the NIT of the next even cycle. The CC checks if the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. Once all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

To read the generated tables, the application must lock the tables to prevent the CC from updating these tables. The locking is initiated by writing a 1 to the even table lock trigger FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding

status information fields SFRA, SFRB in the Sync Frame Counter Register (FR_SFCNTR). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in Figure 54-27.

After reading all the data from the locked tables, the application must unlock the table by writing to the even table lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the Sync Frame Counter Register (FR_SFCNTR) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the Sync Frame Counter Register (FR_SFCNTR).

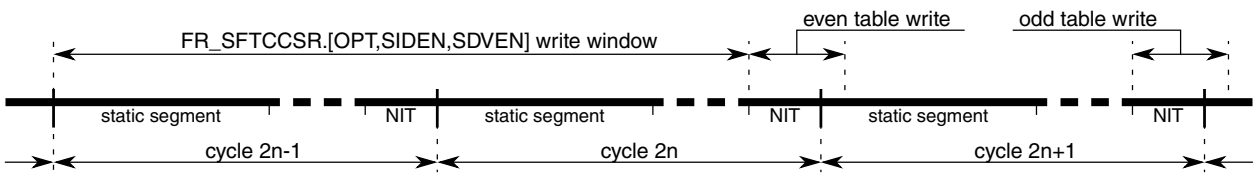


Figure 54-28. Sync Frame Table Trigger and Generation Timing

54.7.12.5 Sync Frame Table Access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in Figure 54-27. During the table write, the application can not lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

54.7.12.5.1 Sync Frame Table Locking and Unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

54.7.13 MTS Generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol MTS in the symbol window on channel A or channel B.

The application can configure the set of communication cycles in which the MTS will be transmitted over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the MTS A Configuration Register (FR_MTSACFR) and MTS B Configuration Register (FR_MTSBCFR).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the MTS A Configuration Register (FR_MTSACFR) or MTS B Configuration Register (FR_MTSBCFR). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$FR\_PSR0[PROTSTATE] = POC:\ normalactive$$

Equation 52

$$FR\_MTSACRF[MTE] = 1$$

Equation 53

$$CYCCNT \,\&\, FR\_MTSACFR[CYCCNTMSK] = FR\_MTSACFR[CYCCNTVAL] \,\&\, FR\_MTSACFR[CYCCNTMSK]$$

Equation 54

The MTS is transmitted over channel B in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$FR\_MTSBCRF[MTE] = 1$$

Equation 55

$$CYCCNT \,\&\, FR\_MTSBCFR[CYCCNTMSK] = FR\_MTSBCFR[CYCCNTVAL] \,\&\, FR\_MTSBCFR[CYCCNTMSK]$$

Equation 56

54.7.14 Key Slot Transmission

Key slot assignment, transmission in POC:startup state, and transmission in POC:normal active state are discussed in this section.

54.7.14.1 Key Slot Assignment

A key slot is assigned to the CC if the key_slot_id field in the Protocol Configuration Register 18 (FR_PCR18) is configured with a value greater than 0 and less or equal to number_of_static_slots in Protocol Configuration Register 2 (FR_PCR2), otherwise no key slot is assigned.

54.7.14.2 Key Slot Transmission in POC:startup

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by FlexRay Communications System Protocol Specification, Version *2.1 Rev A* .

54.7.14.3 Key Slot Transmission in POC:normal active

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in the following table is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (see Message Transmission). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (see Null Frame Transmission).

Table 54-53. Key Slot Frame Type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	normal frame
0	1	normal frame1[1]
1	0	sync frame
1	1	startup frame

1. The frame transmitted has an semantically incorrect header and will be detected as an invalid frame at the receiver.

54.7.15 Sync Frame Filtering

Each received synchronization frame must pass the Sync Frame Acceptance Filter and the Sync Frame Rejection Filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled, i.e. the SFFE control bit in the Module Configuration Register (FR_MCR) is cleared, all received synchronization

frames are considered for clock synchronization. If a received synchronization frame did not pass at least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

54.7.15.1 Sync Frame Acceptance Filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR) and the mask is configured in the Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if the following equations evaluates to true.

$$FR\_MCR[SFFE] = 0$$

Equation 57

$$FID \,\&\, FR\_SFIDAFMR[FMSK] = FR\_SFIDAFVR[FVAL] \,\&\, FR\_SFIDAFMR[FMSK]$$

Equation 58

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

54.7.15.2 Sync Frame Rejection Filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the Sync Frame ID Rejection Filter Register (FR_SFIDRFR). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if the following equations evaluates to true.

$$FR\_MCR[SFFE] = 0$$

Equation 59

$$FID \neq FR\_SFIDRFR[SYNFRID]$$

Equation 60

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

54.7.16 Strobe Signal Support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in the Strobe Signal Mapping table, located in the Register Descriptions section.

54.7.16.1 Strobe Signal Assignment

Each of the strobe signals listed in the Strobe Signal Mapping table, located in the Register Descriptions section, can be assigned to one of the four strobe ports using the Strobe Signal Control Register (FR_STBSCR). To assign multiple strobe signals, the application must write multiple times to the Strobe Signal Control Register (FR_STBSCR) with appropriate settings.

To read out the current settings for a strobe signal with number N, the application must execute the following sequence.

1. Write to FR_STBSCR with WMD = 1 and SEL = N. (updates SEL field only)
2. Read STBCSR.

   The SEL field provides N and the ENB and STBPSEL fields provides the settings for signal N.

54.7.16.2 Strobe Signal Timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay Bus. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a negative clock offset. An example waveform is given in the following figure.

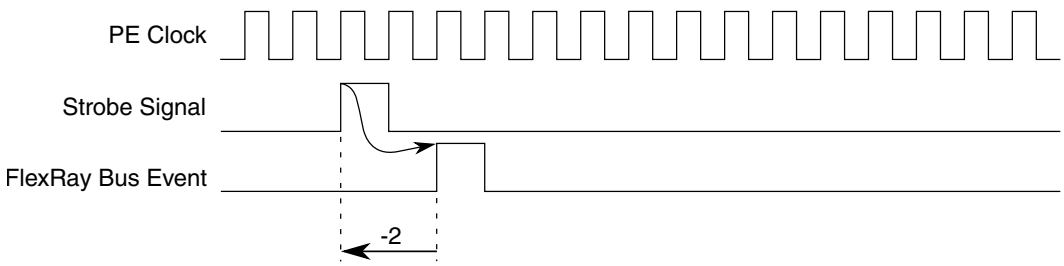


Figure 54-29. Strobe Signal Timing (type = pulse, clk_offset = -2)

Other signals refer to events that occurred on the FlexRay Bus some cycles before the strobe signal is changed. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a positive clock offset. An example waveform is given in the figure below.

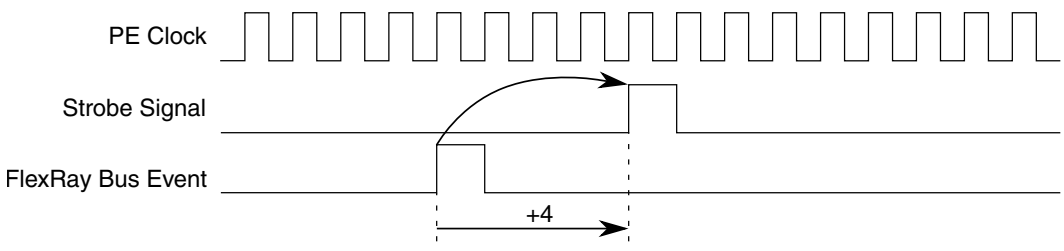


Figure 54-30. Strobe Signal Timing (type = pulse, clk_offset = +4)

54.7.17 Timer Support

The CC provides two timers, which run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In the non-repetitive mode, timer stops if it expires. In repetitive mode, timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

54.7.17.1 Absolute Timer T1

The absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set at the macrotick start event, if the following equations are fulfilled.

$$CYCTR[CTCCNT] \& FR\_TI1CYSR[T1\_CYC\_MSK] = FR\_TI1CYSR[T1\_CYC\_VAL] \& FR\_TI1CYSR[T1\_CYC\_MSK]$$

Equation 61

$$FR\_MTCTR[MTCT] = FR\_TI1MTOR[T1\_MTOFFSET]$$

Equation 62

If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The status bit T1ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T1ST bit is not cleared and the timer is restarted immediately. The T1ST is cleared when the timer is stopped.

The corresponding Interrupt condition (i.e., when the abovementioned equations are fulfilled) also leads to an event out indication on an output port (*fr_evt_tim1*) based on the [TIM1_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

54.7.17.2 Absolute / Relative Timer T2

The timer T2 can be configured to be an absolute or relative timer by setting the T2_CFG control bit in the Timer Configuration and Control Register (FR_TICCR). The status bit T2ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T2ST bit is not cleared and the timer is restarted immediately. The T2ST is cleared when the timer is stopped.

54.7.17.2.1 Absolute Timer T2

If timer T2 is configured as an absolute timer, it has the same functionality timer T1 but the configuration from Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) is used. On expiration of timer T2, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set. If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The corresponding Interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the [TIM2_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

54.7.17.2.2 Relative Timer T2

If the timer T2 is configured as a relative timer, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set, when the programmed amount of macroticks MT[31:0], defined by Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1), has expired since the trigger or restart of timer 2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of MT[31:0] is checked and then decremented. Thus, if the timer is started with MT[31:0] == 0, it expires at the next macrotick start.

The corresponding Interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the [TIM2_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

54.7.18 Slot Status Monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per channel base. The content of the slot status vector is described in the "Slot Status Content" table below. The PE provides the slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in the below figure.

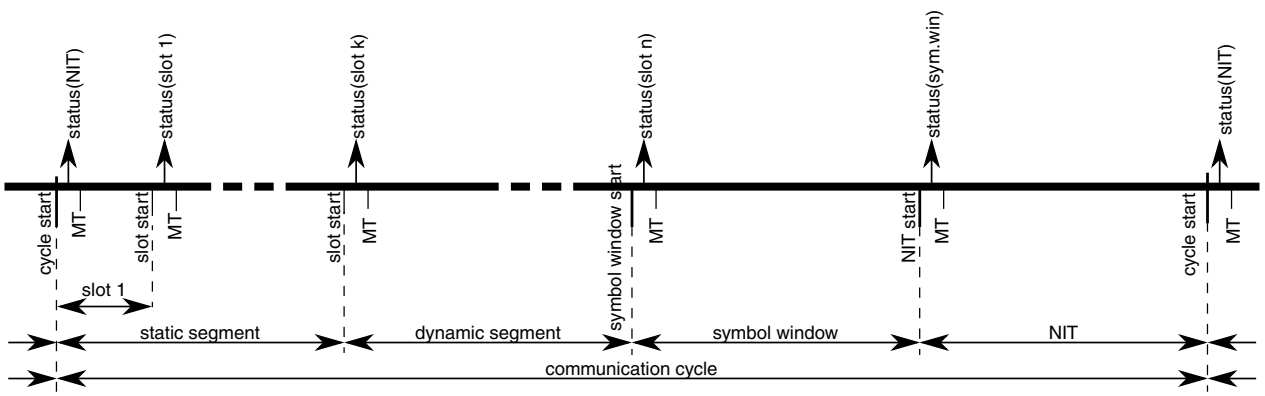


Figure 54-31. Slot Status Vector Update

Note

The slot status for the NIT of cycle n is provided after the start of cycle n+1.

Table 54-54. Slot Status Content

	Status Content
static /dynamic Slot	**slot related status**

Table continues on the next page...

Table 54-54. Slot Status Content (continued)

	Status Content
	vSS!ValidFrame - valid frame received
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!ContentError - content error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	for slots in which the module transmits:
	vSS!TxConflict - reception ongoing while transmission starts
	for slots in which the module does not transmit:
	vSS!TxConflict - reception ongoing while transmission starts
	first valid - channel that has received the first valid frame
	received frame related status
	extracted from
	a)header of valid frame, if vSS!ValidFrame = 1
	b) last received header, if vSS!ValidFrame = 0
	c) set to 0, if nothing was received
	vRF!Header!NFIndicator - Null Frame Indicator (0 for null frame)
	vRF!Header!SuFIndicator - Startup Frame Indicator
	vRF!Header!SyFIndicator - Sync Frame Indicator
Symbol Window	**window related status**
	vSS!ValidFrame - always 0
	vSS!ContentError - content error occurred while receiving
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	vSS!TxConflict - reception ongoing while transmission starts
	received symbol related status
	vSS!ValidMTS - valid Media Test Access Symbol received
	received frame related status
	see static/dynamic slot
NIT	**NIT related status**
	vSS!ValidFrame - always 0
	vSS!ContentError - content error occurred while receiving
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	vSS!TxConflict - always 0
	received frame related status
	see static/dynamic slot

54.7.18.1 Channel Status Error Counter Registers

The two channel status error counter registers, Channel A Status Error Counter Register (FR_CASERCR) and Channel B Status Error Counter Register (FR_CBSERCR), incremented by one, if at least one of four slot status error bits, *vSS!SyntaxError, vSS! ContentError, vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap round after they have reached the maximum value.

54.7.18.2 Protocol Status Registers

The Protocol Status Register 2 (FR_PSR2) provides slot status information about the Network Idle Time NIT and the Symbol Window. The Protocol Status Register 3 (FR_PSR3) provides aggregated slot status information.

54.7.18.3 Slot Status Registers

The eight slot status registers, Slot Status Registers (FR_SSR0–FR_SSR7), can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status related and received frame / symbol related status information, as given in Table 54-54, except of the *first valid* indicator for non-transmission slots.

54.7.18.4 Slot Status Counter Registers

The CC provides four slot status error counter registers, Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the Slot Status Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in the figure below.

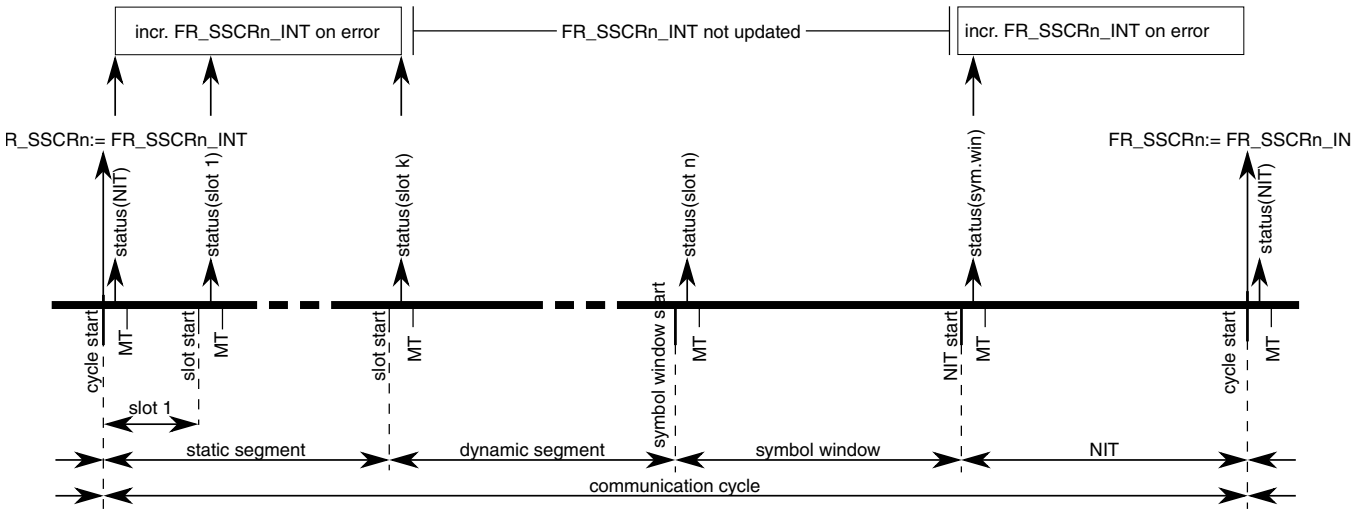


Figure 54-32. Slot Status Counting and FR_SSCRn Update

The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCRn register reflects, in cycle n, the status of the NIT of cycle n-2, and the status of all static slots and the symbol window of cycle n-1.

The increment condition for each slot status counter consists of two parts, the frame related condition part and the slot related condition part. The internal slot status counter FR_SSCRn_INT is incremented if at least one of the conditions is fulfilled:

1. frame related condition:

   (FR_SSCCRn[VFR] | FR_SSCCRn[SYF] | FR_SSCCRn[NUF] | FR_SSCCRn[SUF]) // count on frame condition

   = 1;

   and

   ((~FR_SSCCRn[VFR] | vSS!ValidFrame) & // valid frame restriction

   (~FR_SSCCRn[SYF] | vRF!Header!SyFIndicator) & // sync frame indicator restriction

   (~FR_SSCCRn[NUF] | ~vRF!Header!NFIndicator) & // null frame indicator restriction

   (~FR_SSCCRn[SUF] | vRF!Header!SuFIndicator)) // startup frame indicator restriction

   = 1;

Note

The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus it is required to set the VFR always, whenever count on frame condition is used.

slot related condition:

((FR_SSCCRn[STATUSMASK[3]] & vSS!ContentError) | // increment on content error

(FR_SSCCRn[STATUSMASK[2]] & vSS!SyntaxError) | // increment on syntax error

(FR_SSCCRn[STATUSMASK[1]] & vSS!BViolation) | // increment on boundary violation

(FR_SSCCRn[STATUSMASK[0]] & vSS!TxConflict)) // increment on transmission conflict

= 1;

If the slot status counter is in single cycle mode, i.e. FR_SSCCRn[MCY] = 0, the internal slot status counter FR_SSCRn_INT is reset at each cycle start. If the slot status counter is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset and incremented, until the maximum value is reached.

54.7.18.5 Message Buffer Slot Status Field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in Table 54-54 for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update Sections in Individual Message Buffer Configuration Data.

54.7.19 System Bus Access

This section provides a description of the system bus accesses failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in System Bus Access Failure Types.

The behavior of the CC after the occurrence of a system bus access failure is described in System Bus Access Failure Response.

54.7.19.1 System Bus Access Failure Types

This section describes the two types of system bus access failures.

54.7.19.1.1 System Bus Illegal Address Access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions which are treated as a system bus illegal address access:

- The system bus subsystem detects an CC access to an illegal system memory address.
- The CC detects the usage of an data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (see CHI LRAM Error Response after CC Read).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the CHI Error Flag Register (FR_CHIERFR).

54.7.19.1.2 System Bus Access Timeout

A system bus access timout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (see Configure System Memory Access Time-Out Register (FR_SYMATOR)).

If a system bus access timout is detected, the CC sets the SBCF_EF flag in the CHI Error Flag Register (FR_CHIERFR).

54.7.19.2 System Bus Access Failure Response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the Module Configuration Register (FR_MCR).

54.7.19.2.1 Continue after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is 0, the CC will continue its operation after the occurrence of the system bus access failure, but will not generate any system bus accesses until the start of the next communication cycle.Since no

data are read from or written to the FlexRay memory area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated with the remaining header and frame data are replaced by all zeros. Depending on the point in time this can affect the PPI bit, the Header CRC, the Payload Length in case of an dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted and received, except for the key slot, where a sync or startup null-frame is transmitted, if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

54.7.19.2.2 Freeze after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is set to 1, the CC will go into the freeze mode immediately after the occurrence of one of the system bus access failures.

54.7.20 Interrupt Support

The CC provides 172 individual interrupt sources and five combined interrupt sources.

54.7.20.1 Individual Interrupt Sources

Message buffer interrupts are discussed in this section.

54.7.20.1.1 Message Buffer Interrupts

The CC provides 128 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSRn[MBIF] and an interrupt enable bit FR_MBCCSRn[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

54.7.20.1.2 FIFO Interrupts

The CC provides 2 FIFO interrupt sources.

Each of the 2 FIFO provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the Global Interrupt Flag and Enable Register (FR_GIFER) if the corresponding Receive FIFO fill level exceeds the defined watermark.

54.7.20.1.3 Wakeup Interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

54.7.20.1.4 Protocol Interrupts

The CC provides 25 interrupt sources for protocol related events. For details, see Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1). Each interrupt source has its own interrupt enable bit.

54.7.20.1.5 CHI Interrupts

The CC provides 16 interrupt sources for CHI related error events. For details, see CHI Error Flag Register (FR_CHIERFR). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

54.7.20.2 Combined Interrupt Sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

54.7.20.2.1 Receive Message Buffer Interrupt

The Receive Message Buffer Interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

54.7.20.2.2 Transmit Message Buffer Interrupt

The Transmit Message Buffer Interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

54.7.20.2.3 Protocol Interrupt

The Protocol Interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set..

54.7.20.2.4 CHI Interrupt

The CHI Interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

54.7.20.2.5 Module Interrupt

The Module Interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

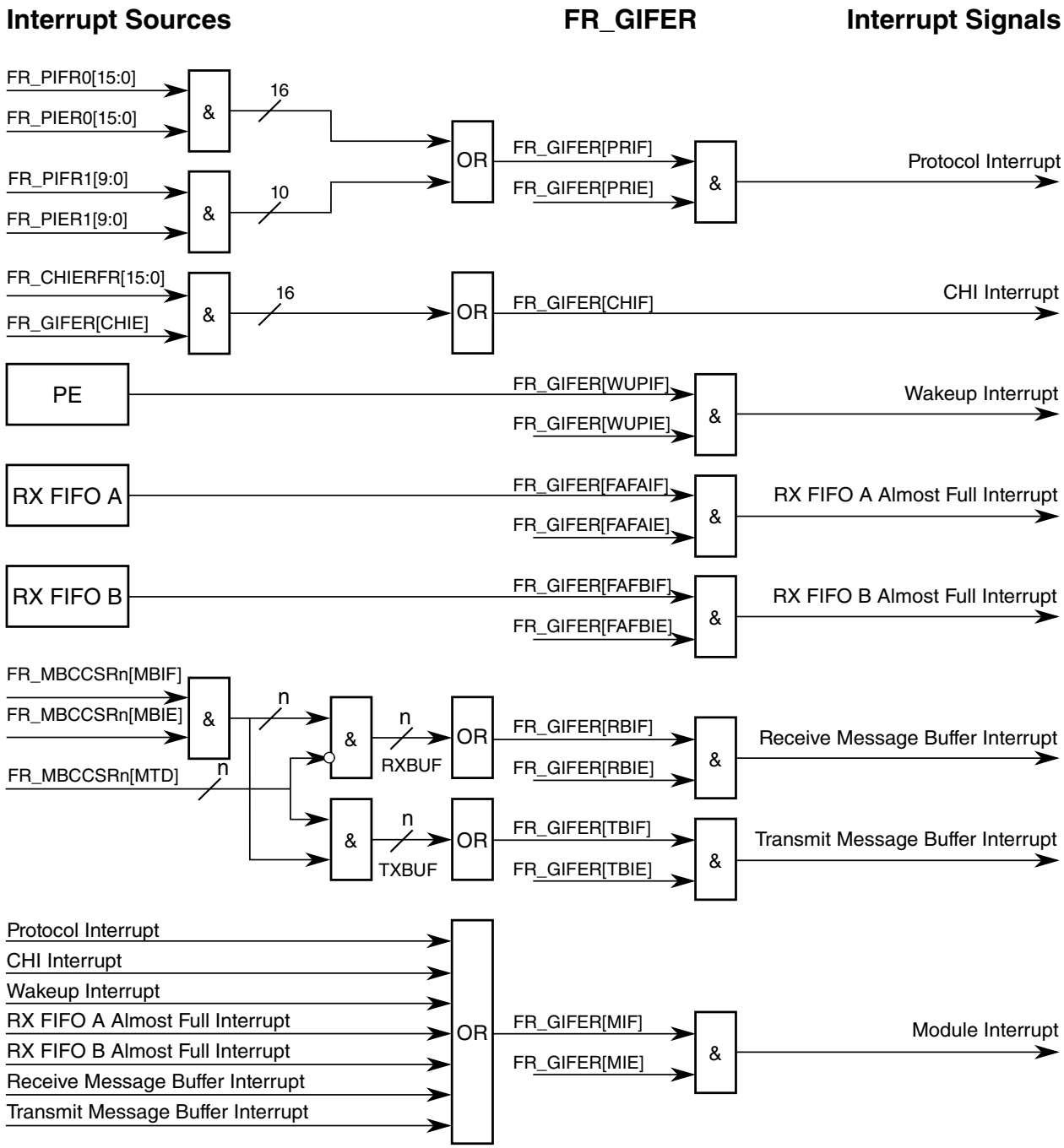


Figure 54-33. Scheme of FR_GIFER interrupt signal generation

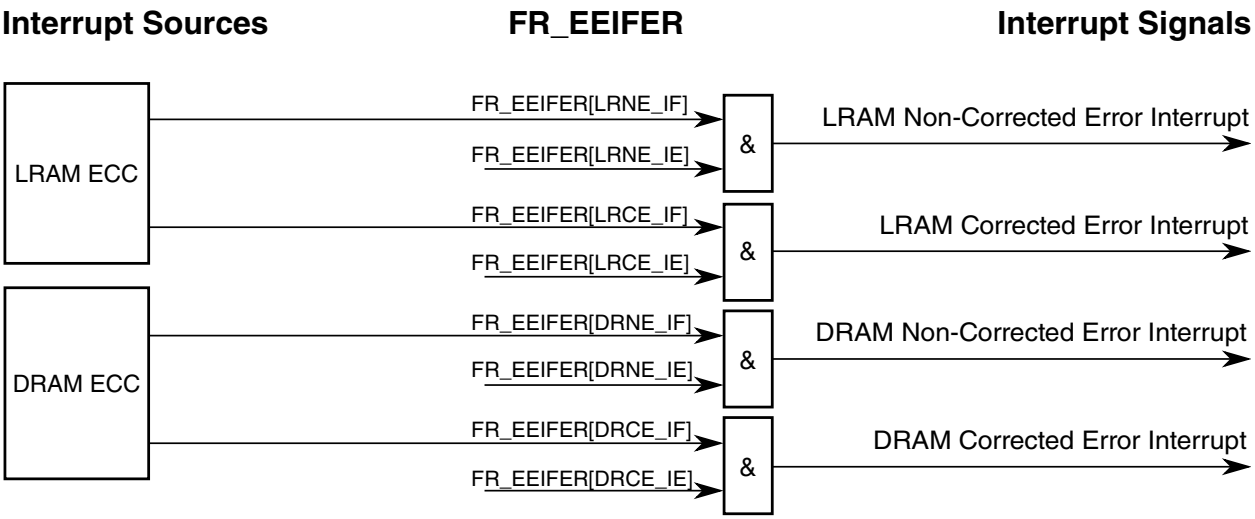


Figure 54-34. Scheme of FR_EEIFER interrupt signal generation

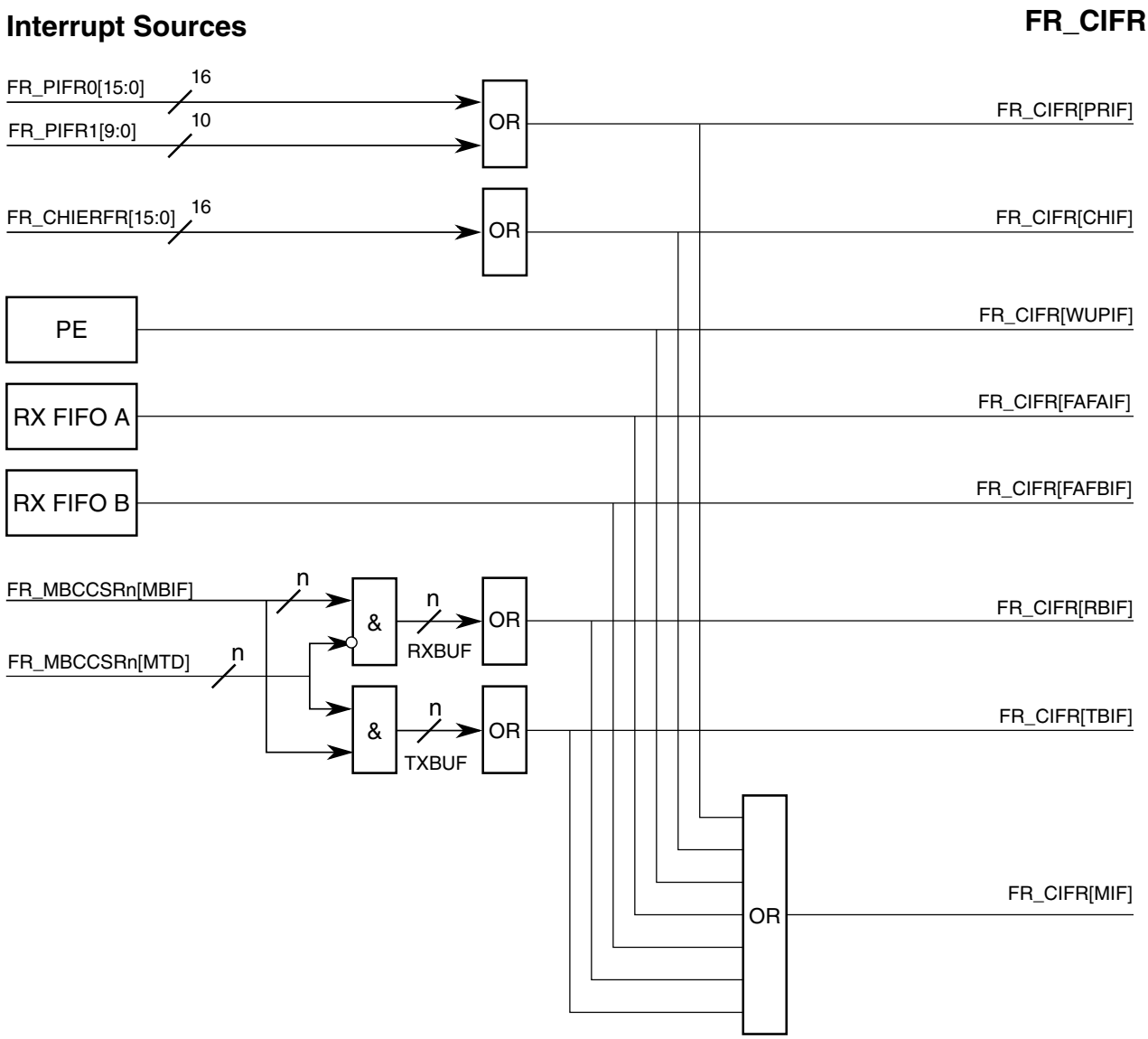


Figure 54-35. Scheme of FR_CIFR flags generation

54.7.21 Lower Bit Rate Support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset*. The application configures the FlexRay channel bit rate by setting the BITRATE field in the Module Configuration Register (FR_MCR). The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings and related protocol parameter values are shown in the "FlexRay Channel Bit Rate Control" table below.

Table 54-55. FlexRay Channel Bit Rate Control

FlexRay Channel Bit Rate [Mbit/s]	FR_MCR[BITRATE]	*pdMicrotick [ns]*	*gdSampleClockPeriod [ns]*	*pSamplesPerMicrotick*	*cSamplesPerBit*	*cStrobeOffset*
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note

The bit rate of 8 Mbit/s is not defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

54.7.22 PE Data Memory (PE DRAM)

The PE Data Memory (PE DRAM) is 128 word, 16-bit wide memory with byte access, which contains the program data of the PE internal CPU. The PE DRAM is divided into two banks, 8-bit each. The memory data [7:0] are assigned to BANK0, the memory data [15:8] are assigned to BANK1.

Table 54-56. PE DRAM Layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
…		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via PE DRAM Access Register (FR_PEDRAR) and PE DRAM Access Register (FR_PEDRAR). This functionality is used to check the memory error detection.

54.7.22.1 PE DRAM Read Access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;

   // INST=0x5; ADDR=070

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA];

   // read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause an response delay with a maximum of 1000 PE clock cycle (25us).

54.7.22.2 PE DRAM Write Access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;

   // write value to be written into data register

2. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

3. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

4. val = FR_PEDRDR[DATA];

   // read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may causes an response delay with a maximum of 1000 PE clock cycle (25us).

If the conditions given in PE DRAM Write Access Limitations are fulfilled, the data provided in PE DRAM Access Regsiter (FR_PEDRAR) are written into the PE DRAM, read back in the next clock cycle and stored into the PE DRAM Access Register (FR_PEDRAR). Otherwise, data are not written into the PE DRAM and 0x0000 is stored into the PE DRAM Access Register (FR_PEDRAR).

54.7.22.3 PE DRAM Write Access Limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not *in POC:default config* state, only PE DRAM address 0x70 is writable.

54.7.23 CHI Lookup-Table Memory (CHI LRAM)

The CHI Lookup-Table Memory (CHI LRAM) is an CHI internal memory which contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or 6 data field offsets are contained in one memory row. The CHI LRAM is divided into 6 memory BANKs.

Table 54-57. CHI LRAM Layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
…						
0x3F	FR_MBIDXR127	FR_MBFIDR127	FR_MBCCFR127	FR_MBIDXR126	FR_MBFIDR126	FR_MBCCFR126
0x40	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
…						
0x55	FR_MBDOR131	FR_MBDOR130	FR_MBDOR129	FR_MBDOR128	FR_MBDOR127	FR_MBDOR126
0x56	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

54.7.23.1 CHI LRAM Read and Write Access

The CHI LRAM is accessed by the application via regular register read and write accesses.

54.7.24 Memory Content Fault Detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses an standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDDED) . The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SECDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

54.7.24.1 Memory Error Types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as an special case. The FlexRay module detects and indicates the memory errors as shown in the "Detected Memory Error Types" table below. The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row, and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error free bank has been read.

Table 54-58. Detected Memory Error Types

Memory	Priority	Memory Data	Indication
CHI LRAM	0 (highest)	All Zero's	Non-Corrected Error
PE DRAM			Non-Corrected Error
CHI LRAM		All One's	Non-Corrected Error
PE DRAM			
CHI LRAM	1 (lowest)	One Bit Flipped	Non-Corrected Error
PE DRAM			Corrected Error

Table continues on the next page...

Table 54-58. Detected Memory Error Types (continued)

Memory	Priority	Memory Data	Indication
CHI LRAM		Two Bits Flipped	Non-Corrected Error
PE DRAM			
CHI LRAM		Three or more Bits Flipped	one out of {No error, Non-Corrected Error}, defined by coding given in CHI LRAM Checkbits and CHI LRAM Syndrome
PE DRAM			one out of {No error, Corrected Error, Non-Corrected Error}, defined by coding given in PE DRAM Checkbits and PE DRAM Syndrome

54.7.24.2 Memory Error Reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

For each of the two memories exists two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the ECC Error Interrupt Flag and Enable Register (FR_EEIFER) is set.

- *If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.*
- *If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.*

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to above the Error indications (corrected / non corrected) along with the failing address is sent out from the module for external error logging and corrective actions by the Software.

54.7.24.2.1 PE DRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in Table 54-58. This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 54-59. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4[1]	X	X	X	X	X	X	X	X	X	X	X	X
3[2]	-	-	-	-	X	X	X	X	-	-	-	-
2	-	-	-	-	X	-	-	-	X	X	X	-
1	-	-	-	-	-	X	X	-	X	X	-	X
0	-	-	-	-	-	X	-	X	X	-	X	X

1. The checkbit CODE[4] is set to 1 if and only if there is a even number of 1's in columns with X.
2. The checkbits CODE[3]… CODE[0] are set to 1 if and only if there is a odd number of 1's in all columns with X.

This coding of the checkbit ensures that neither 0x000 nor 0xFFF are valid code words and written into the memory.

54.7.24.2.2 PE DRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in the following table.

Table 54-60. FR_EERCR[CODE] PE DRAM Syndrome Coding

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (Never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data!= 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected Error (Parity Bit 0)
0x0	0x2	Corrected Error (Parity Bit 1)
0x0	0x3	Corrected Error (Data Bit 0)
0x0	0x4	Corrected Error (Parity Bit 2)
0x0	0x5	Corrected Error (Data Bit 1)
0x0	0x6	Corrected Error (Data Bit 2)
0x0	0x7	Corrected Error (Data Bit 3)
0x0	0x8	Corrected Error (Parity Bit 3)
0x0	0x9	Corrected Error (Data Bit 4)

Table continues on the next page...

Table 54-60. FR_EERCR[CODE] PE DRAM Syndrome Coding (continued)

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x0	0xA	Corrected Error (Data Bit 5)
0x0	0xB	Corrected Error (Data Bit 6)
0x0	0xC	Corrected Error (Data Bit 7)
0x0	0xD-0xF	Non-Corrected Error
0x1	0x1-0xF	Non-Corrected Error

54.7.24.2.3 CHI LRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the table below. This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 54-61. CHI LRAM checkbits coding

CODE[1]	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	X	X	X	X	X	X	X	-	-	-	-
2	X	X	-	-	-	X	X	X	X	-	-	-	X	X	X	-
1	-	-	X	X	-	X	X	-	-	X	X	-	X	X	-	X
0	X	-	X	-	X	X	-	X	-	X	-	X	X	-	X	X

1. The checkbit CODE[n] is set to 1 if and only if there is a odd number of 1's in all columns with X.

54.7.24.2.4 CHI LRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the next table.

Table 54-62. FR_EERCR[CODE] CHI LRAM Syndrome Coding

FR_EERCR[CODE]	Description
0x00	No Error (Never appears in error report registers)
0x01-0x1F	Non Corrected Error

54.7.24.3 Memory Error Response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

In case of the detection of a *corrected* memory error, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

54.7.24.3.1 CHI LRAM Error Response after CC Read

When the CC is out of the *POC:default config* state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access the error response of the module depends from LRAM location where the error occurred.

- *If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.*

  If the affected message buffer is a tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is a rx message buffer, no frame will be received to this message buffer in the next slot.

- *If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs w.r.t. payload access is performed for the complete frame.*

54.7.24.3.2 CHI LRAM Error Response after Application Read

The application can read the content of the CHI LRAM via reading the FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn, FR_MBDORn, and FR_LEETRn registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read and continues its normal operation.

54.7.24.3.3 PE DRAM Error Response after CC Read

If the CC detects an non-corrected memory error during internal read of program data which is contained in PE DRAM, this is considered as an fatal protocol error and the module enters the protocol freeze state immediately.

54.7.24.3.4 PE DRAM Error Response after Application Read in *POC:default config* state

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered as an fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

54.7.24.3.5 PE DRAM Error Response after Application Read out of *POC:default config*

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address, and the protocol is not in the *POC:default config* state, this error is not considered as an fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

54.7.25 Memory Fault Injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

The error injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).

When the error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

54.7.25.1 CHI LRAM Error Injection

The following sequence describes an memory error injection sequence for the CHI LRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content. The content of the CHI LRAM is described in Table 54-57.

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see Memory Fault Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode
3. FR_EEIAR[MID]:= 1;

   // select CHI LRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;

   // select bank for error injection; I_BANK = {0,1,2,3,4,5}
5. FR_EEIAR[ADDR]:= I_ADDR;

   // select address for error injection; I_ADDR <= 0x 56
6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access:

If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;

If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;

If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;

If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;

If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;

If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;

// write DATA to the defined injection bank and injection address (see Table 54-57).

54.7.25.2 PE DRAM Error Injection

The following sequence describes an memory error injection sequence for the PE DRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see PE DRAM Error Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality

2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode

3. FR_EEIAR[MID]:= 0;

   // select PE DRAM for error injection

4. FR_EEIAR[BANK]:= I_BANK;

   // define bank for error injection; I_BANK = {0,1}

5. FR_EEIAR[ADDR]:= I_ADDR;

   // define address for error injection; I_ADDR <= 0x7F

6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern

7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern

8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access (e.g. I_ADDR=0x70):

1. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA]; |

   // get read back PE DRAM data

Note

The write access to the PE DRAM triggers an subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

54.7.26 StopWatch function

FlexRay supports a StopWatch function to facilitate OS synchronization to the FlexRay network. This provides a capability to capture the time difference in system clocks between the operating system timer tick and the FlexRay time base. This allows the software synchronization handler to adjust the OS clock to sync with the FR macrotick and are fulfilled.

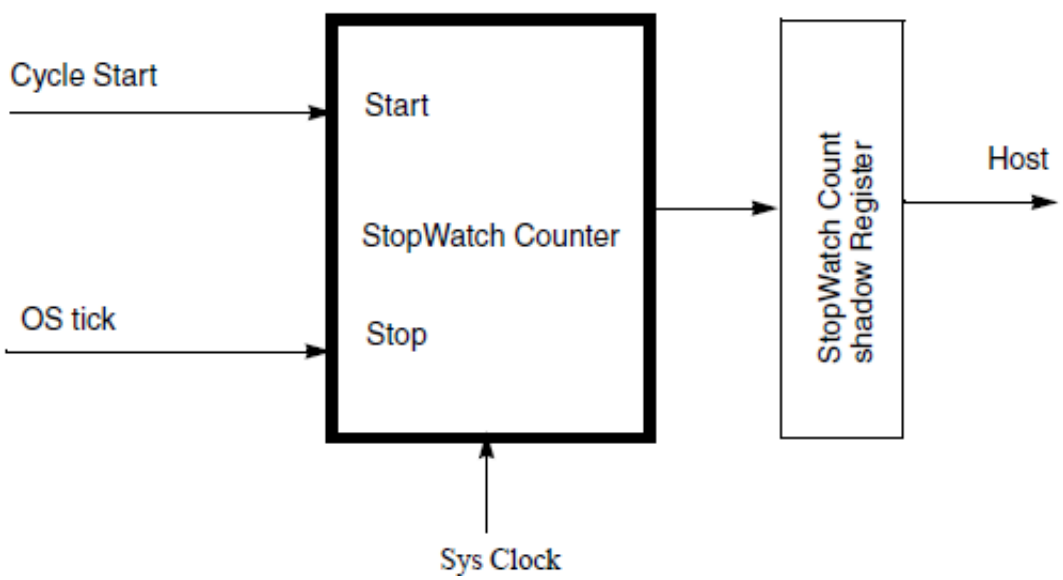


Figure 54-36. StopWatch counter logic

As shown below, there is a 32 bit stopwatch counter that resets and starts with each cycle start. This counter runs on the system clock and increments every cycle until an OS tick event is received, after which the counter halts and the count value is transferred to a shadow register. The shadow register (Stop Watch Count Register (FR_STPWR)) holds the counter value till the host reads all the 32 bits, after which it automatically gets cleared. The StopWatch logic is controlled through a control bit STPW_EN in the FR_PEOER (Protocol Event Output Enable and StopWatch Control Register (FR_PEOER). The counter runs only when this bit is set. The following figure shows the actual timing of the stopwatch counter logic operation.

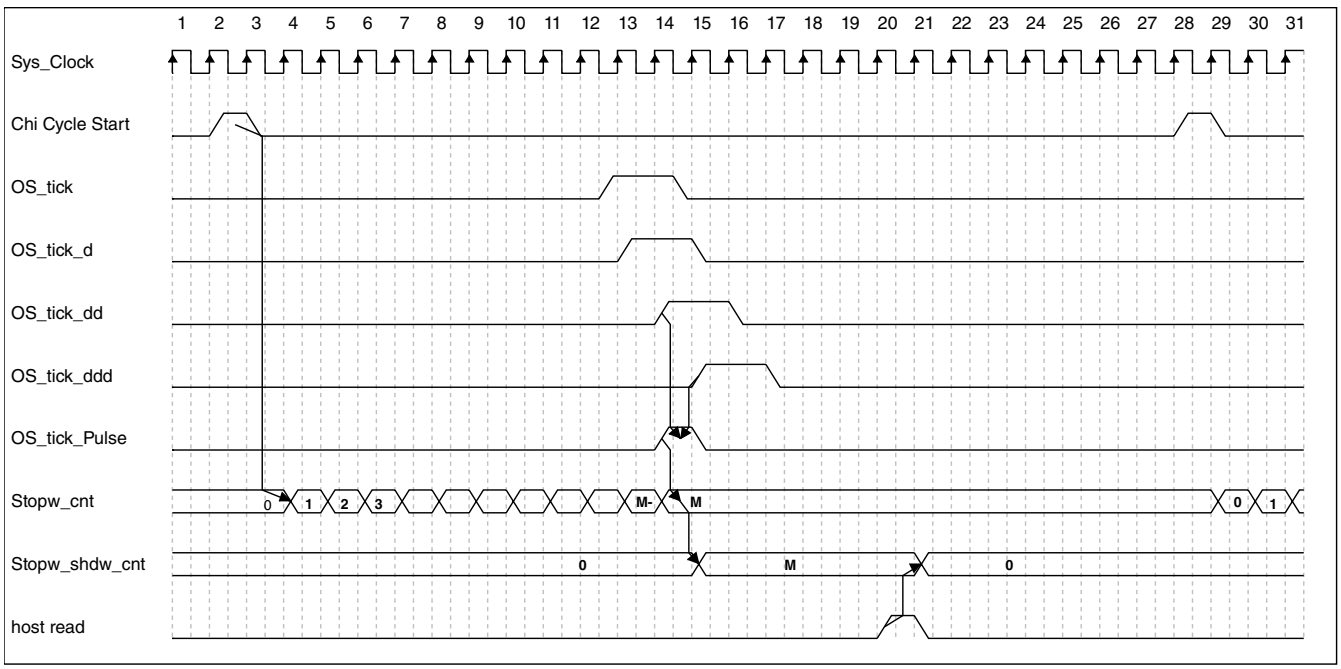


Figure 54-37. StopWatch function timing diagram

NOTE

‘Host read’ implies that all 32 bits have been read out completely. Any partial read (i.e for 8/16 bits access) will not clear the StopWatch count register.

NOTE

If cycle start and OS tick occurs simultaneously, timer will remain at 0. But OS_tick has a 2 stage synchronizer and cycle_start has a single stage synchronizer into chi clock domain. So simultaneously here means OS_tick_pulse and chi_cycle_start occurring at the same cycle.

54.8 Application Information

Module operation is discussed in this section.

54.8.1 Module Configuration

This section describes essential parts of the module configuration.

54.8.1.1 Configure System Memory Access Time-Out Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in System Memory Access Time-Out Register (FR_SYMATOR) and the CHI clock frequency f_{CHI} in MHz fulfill this equation[1] :

$$0 \le \mathrm{SYMATOR}[\mathrm{TIMEOUT}] \le \lfloor 0.45 \bullet f_{CHI} - 8 \rfloor$$

Equation 63

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI}.

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in the table below.

Table 54-63. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
>= 18 MHz	0	>= 100 MHz	<= 37
>= 23 MHz	<= 2	>= 120 MHz	<= 46
>= 27 MHz	<= 4	>= 140 MHz	<= 55
>= 32 MHz	<= 6	>= 160 MHz	<= 64
>= 60 MHz	<= 19	>= 180 MHz	<= 73
>= 80 MHz	<= 28	>= 200 MHz	<= 82

1. See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

54.8.1.1.1 System Bus Wait State Constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single channel configurations and if the sync frame table generation functionality is *not* used (FR_SFTCCSR[SDVEN,SIDEN] = 0) no timeout will be detected if less than 2 × SYMATOR[TIMEOUT]+1 wait states will be seen on the system bus for each system bus access.

For dual channel configurations, or if the sync frame table generation functionality is used, no timeout will be detected if less than SYMATOR[TIMEOUT]-1 wait states will be seen on the system bus for each system bus access.

54.8.1.2 Configure Data Field Offsets

The data field offsets are located in the Message Buffer Data Field Offset Registers (FR_MBDORn) and Receive FIFO Start Data Offset Register (FR_RFSDOR). The application has to configure the data field offset values for all message buffers which are used.

When the module is enabled, the initilization value of the FR_MBDORn[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (see System Bus Illegal Address Access).

54.8.2 Initialization Sequence

This section describes the required steps to initialize the CC. The first subsection describes the steps required after a module reset, the second section describes the steps required after preceding shutdown of the CC.

54.8.2.1 Module Initialization

This section describes the module related initialization steps after a system reset.

1. Configure CC.
    a. configure the control bits in the Module Configuration Register (FR_MCR)
    b. configure the system memory base address in System Memory Base Address Register (FR_SYMBADR)
2. Enable the CC.

a. write 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

The CC now enters the Normal Mode. The application can commence with the protocol initialization described in Protocol Initialization.

54.8.2.2 Protocol Initialization

This section describes the protocol related initialization steps.

1. Configure the Protocol Engine.
   a. issue CONFIG command via Protocol Operation Control Register (FR_POCR)
   b. wait for *POC:config* in Protocol Status Register 0 (FR_PSR0)
   c. configure the FR_PCR0,…, FR_PCR30 registers to set all protocol parameters
2. Configure the Message Buffers and FIFOs.
   a. set the number of message buffers used and the message buffer segmentation in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)
   b. define the message buffer data size in the Message Buffer Data Size Register (FR_MBDSR)
   c. configure each message buffer by setting the configuration values in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn), Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), Message Buffer Index Registers (FR_MBIDXRn)
   d. configure the FIFOs
   e. issue CONFIG_COMPLETE command via Protocol Operation Control Register (FR_POCR)
   f. wait for *POC:ready* in Protocol Status Register 0 (FR_PSR0)

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

54.8.2.3 CHI LRAM Initialization

The initialization of the CHI LRAM is performed by the CC when it leaves the Disabled Mode. The unitization runs for 87 CHI clock cycles. All fields in the FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBDORn, and LEETRn registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

54.8.2.4 PE DRAM Initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 µs, and will delay the state transition from *POC:default config* into *POC:config*.

54.8.3 Memory Fault Injection out of POC:default config

This section provides information for application driven memory fault injection out if *POC:default config*. The CC provides means to inject memory faults from the application without any impacts to the internal protocol operation of the CC.

54.8.3.1 CHI LRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the LRAM ECC Error Test Registers (FR_LEETRn). Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independent from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in CHI LRAM Error Injection.

54.8.3.2 PE DRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only one PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM location to inject memory errors into the PE DRAM. The injection sequence is described in PE DRAM Error Injection.

54.8.4 Shut Down Sequence

This section describes a secure shut down sequence to stop the CC gracefully. The main targets of this sequence are

- *finish all ongoing reception and transmission*
- *do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication*

For a graceful shutdown the application shall perform the following tasks:

1. Disable all enabled message buffers.
   a. repeatedly write 1 to FR_MBCCSRn[EDT] until FR_MBCCSRn[EDS] == 0.
2. Stop Protocol Engine.
   a. issue HALT command via Protocol Operation Control Register (FR_POCR)
   b. wait for *POC:halt* in Protocol Status Register 0 (FR_PSR0)

54.8.5 Number of Usable Message Buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR), a configured mini-slot length *gdMinislot*, and a configured nominal macrotick length *gdMacrotick*[1].

Additional constraints for the minimum CHI clock frequency are given in Controller Host Interface Clocking.

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of slot and must be finished before the start of the next slot.

The shortest FlexRay slot is a corrected empty dynamic slot. An corrected empty dynamic slot is a mini-slot and consists of gdMinislot corrected macroticks with a duration of *gdMacrotick*. The minimum duration of an corrected macrotick is $gdMacrotick_{min}$ = 39 µT. This results in a minimum length of an correct slot

$$\Delta_{slotmin} = 39 \bullet pdMicrotick \bullet gdMinislot$$

Equation 64

1. See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers FR_MBSSUTR[LAST_MB_UTIL] + 1 and for a given CHI clock frequency f_{chi}, this results in a search duration of

$$\Delta_{search} = \frac{1}{f_{chi}} \bullet (FR\_MBSSUTR[LAST\_MB\_UTIL] + 27)$$

Equation 65

The message buffer search must be finished within one slot which requires fulfillment of this equation::

$$\Delta_{search} \leq \Delta_{slotmin}$$

Equation 66

This results in the formula given below which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

$$f_{chi} \geq \frac{(FR\_MBSSUTR[LAST\_MB\_UTIL]+27)}{39 \bullet pdMicrotick \bullet gdMinislot}$$

Equation 67

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR) set to 127 is given in the following table.

Table 54-64. Minimum fchi [MHz] examples (128 message buffers used)

pdMicrotick [ns]	*gdMinislot*					
	2	**3**	**4**	**5**	**6**	**7**
25.0	79.5	53	39.8	31.8	26.5	22.8
50.0	39.8	26.5	19.9	15.9	13.3	11.4

Note

If the minimum CHI frequency is not met the CHIERFR[MBS_EF] flag is set. Refer to the CHI Error Flag Register (FR_CHIERFR) for details.

54.8.6 Protocol Control Command Execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of the FR_POCR Field Descriptions table, located in the Register Descriptions section, by writing the command to the POCCMD field of the Protocol Operation Control Register (FR_POCR). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in the next table. If the current protocol state does not allow the execution of this protocol command (see POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by the table below. If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 54-65. Protocol Control Command Priorities

Protocol Command	Priority	Interrupted By	Cleared and Terminated By
FREEZE	(highest) 1	none	
READY	2		
CONFIG_COMPLETE	3		
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLET, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE,fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(lowest) 10		FREEZE, READY, CONFIG_COMPLETE,fatal protocol error

54.8.7 Message Buffer Search On Simple Message Buffer Configuration

This sections describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

54.8.7.1 Simple Message Buffer Configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot *S*. The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number *t* and has following configuration

Table 54-66. Transmit Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*t*	MTD	1	transmit buffer
FR_MBCCFR*t*	MTM	0	event transition mode
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000011	cycle set = {4n} = {0,4,8,12,…}
	CCFVAL	000000	
FR_MBFIDR*t*	FID	*S*	assigned to slot S

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSRt[CMT] and the lock bit FR_MBCCSRt[LCKS].

The receive message buffer has the message buffer number r and has following configuration

Table 54-67. Receive Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*r*	MTD	0	receive buffer
FR_MBCCFR*r*	MTM	-	n/a
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000001	cycle set = {2n} = {0,2,4,6,…}

Table continues on the next page...

Table 54-67. Receive Buffer Configuration (continued)

Register	Field	Value	Description
	CCFVAL	000000	
FR_MBFIDR*r*	FID	*S*	subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSRt[EDS] = 1 and FR_MBCCSRr[EDS] = 1)

Note

The cycle set {4n+2} = {2,6,10,…} is assigned to the receive buffer only.

The cycle set {4n} = {0,4,8,12,…} is assigned to both buffers.

54.8.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot *S* in the *static* segment.

The configuration of a transmit buffer for a static slot *S* assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot *S*. In any case, the result of the message buffer search will be the transmit message buffer *t*. The receive message buffer *r* will not be found, no reception is possible.

54.8.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot *S* in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

54.8.7.3.1 Transmit Data Not Available

If transmit data are *not available*, i.e. the transmit buffer is not committed FR_MBCCSR*t*[CMT]=0 and/or locked FR_MBCCSR*t*[LCKS]=1,

1. for the cycles in the set {4n}, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in the following figure.

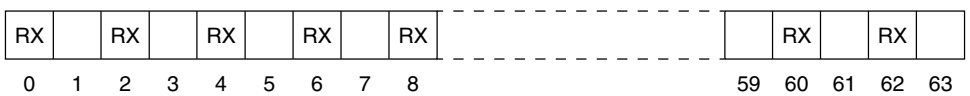


Figure 54-38. Transmit Data Not Available

54.8.7.3.2 Transmit Data Available

If transmit data are *available*, i.e. the transmit buffer is committed FR_MBCCSR*t*[CMT]=1 and not locked FR_MBCCSR*t*[LCKS]=0,

1. for the cycles in the set {4n}, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in the following figure.

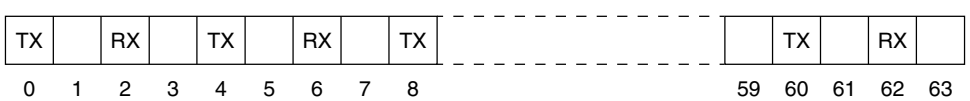


Figure 54-39. Transmit Data Available

Chapter 55
Inter-Integrated Circuit (I2C)

55.1 Overview

This chapter describes the Inter-Integrated Circuit (I^2C) bus module implemented on this chip and presents the following topics:

- Introduction to I^2C
- External signal descriptions
- Memory map and register definition
- Functional description
- Initialization/application information

55.2 Introduction to I^2C

This section presents the following topics:

- Definition: I^2C module
- Advantages of the I^2C bus
- Module block diagram
- Features
- Modes of operation
- Definition: I^2C conditions

55.2.1 Definition: I²C module

The I²C module is a functional unit that provides a two-wire— serial data (SDA) and serial clock (SCL) — bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

55.2.2 Advantages of the I²C bus

The synchronous, multiple-master two-wire I²C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously
- Does not require an external address decoder

55.2.3 Module block diagram

The following figure shows a block diagram of the I²C module.

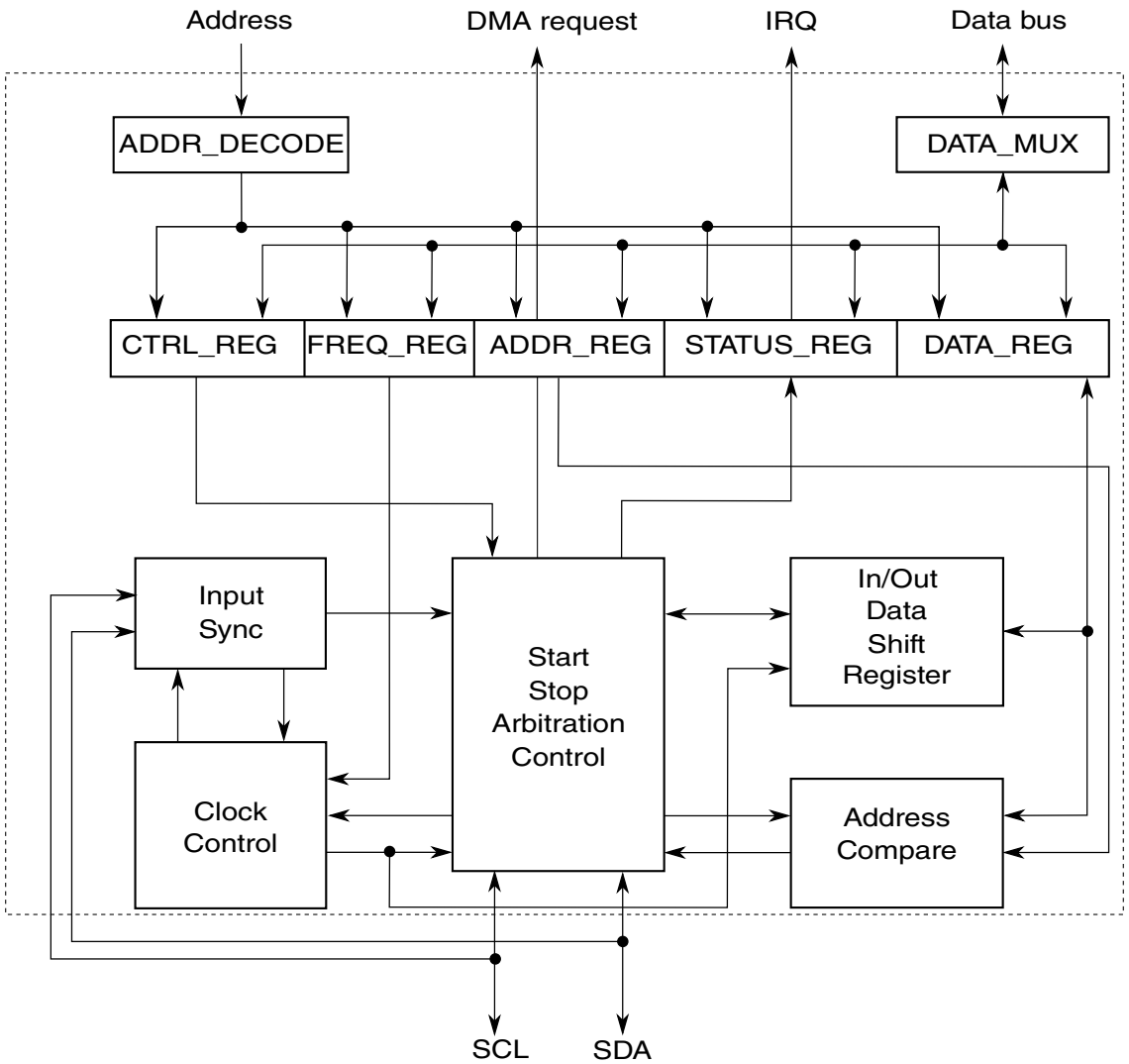


Figure 55-1. I²C block diagram

55.2.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard[1]
- Operating speeds
  - Up to 100kbps in Standard Mode
  - Up to 400kbps in Fast Mode

1. Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

  - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
  - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400pF

55.2.5 Modes of operation

The I^2C module supports the chip modes described in the following table.

Table 55-1. Chip modes supported by the I^2C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I^2C module	The I^2C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. See IPG STOP mode.
IPG DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging	See IPG DEBUG mode.

In addition to chip modes, the I^{2}C module has several module-specific modes. These are described in the following table.

Table 55-2. Module-specific modes supported by the I^{2}C module

Module mode	Description	Important notes
Master mode	The I^{2}C module is the driver of the SDA line.	• Do not use the I^{2}C module's slave address as a calling address. • The I^{2}C module cannot be a master and a slave simultaneously.
Slave mode	The I^{2}C module is not the driver of the SDA line.	• Enable the I^{2}C module before a START condition from a non-I^{2}C master is detected. • By default the I^{2}C module performs as a slave receiver.

55.2.6 Definition: I^{2}C conditions

The following table shows the I^{2}C-specific conditions defined for the I^{2}C module.

Table 55-3. I^{2}C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in the following figure.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in the following figure.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

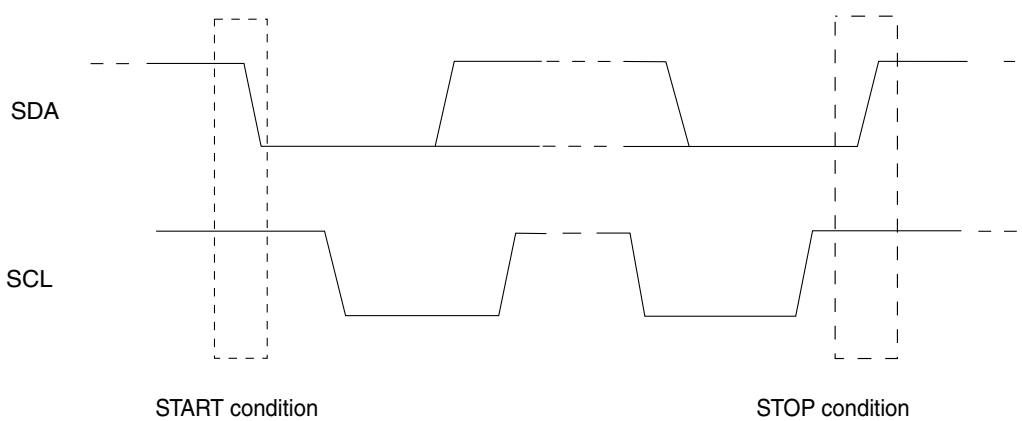


Figure 55-2. START and STOP conditions

55.3 External signal descriptions

This section presents the following topics:

- Signal overview
- Detailed external signal descriptions

55.3.1 Signal overview

The I^2C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, see the data sheet for this chip.

55.3.2 Detailed external signal descriptions

The SDA and SCL signals are described in the following table.

Table 55-4. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I^2C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I^2C bus specification

55.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^2C module. It presents the following topics:

- Register accessibility
- Register figure conventions

- I2C Bus Address Register (I2C_IBAD)
- I2C Bus Frequency Divider Register (I2C_IBFD)
- I2C Bus Control Register (I2C_IBCR)
- I2C Bus Status Register (I2C_IBSR)
- I2C Bus Data I/O Register (I2C_IBDR)
- I2C Bus Interrupt Config Register (I2C_IBIC)
- I2C Bus Debug Register (I2C_IBDBG)

55.4.1 Register accessibility

Address location 0x0007 is a reserved location, but access to this location will not generate any bus error.

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries.

As an example, the IBFD is at address offset 0x01, and can be accessed in any of the following ways:

- 8-bit R/W access of address offset 0x0001
- Second byte of 16-bit R/W access of address offset 0x0000
- Second byte of 32-bit R/W access of address offset 0x0000

The IBFD register cannot be accessed by a 16- or 32-bit RW operation at address offset 0x0001 because those operations require an address aligned to a 16- or 32-bit boundary.

55.4.2 Register figure conventions

The register figures show the field structure using the conventions in the following figure.

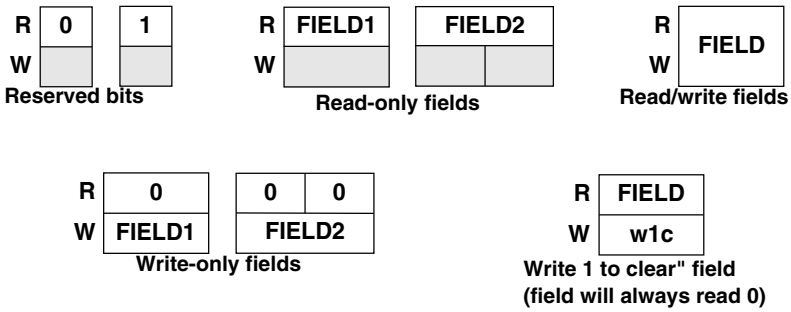


Figure 55-3. Register figure convention

The memory map for the I^{2}C module is given below. The total address for each register is the sum of the base address for the I^{2}C module and the address offset for each register.

I2C memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	I2C Bus Address Register (I2C_IBAD)	8	R/W	00h	55.4.3/2694
1	I2C Bus Frequency Divider Register (I2C_IBFD)	8	R/W	00h	55.4.4/2695
2	I2C Bus Control Register (I2C_IBCR)	8	R/W	80h	55.4.5/2695
3	I2C Bus Status Register (I2C_IBSR)	8	R/W	80h	55.4.6/2697
4	I2C Bus Data I/O Register (I2C_IBDR)	8	R/W	00h	55.4.7/2698
5	I2C Bus Interrupt Config Register (I2C_IBIC)	8	R/W	00h	55.4.8/2699
6	I2C Bus Debug Register (I2C_IBDBG)	8	R/W	00h	55.4.9/2700

55.4.3 I2C Bus Address Register (I2C_IBAD)

This register contains the address the I^{2}C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	ADR							0
Write								
Reset	0	0	0	0	0	0	0	0

I2C_IBAD field descriptions

Field	Description
0–6 ADR	Slave Address. Specific slave address to be used by the I^{2}C Bus module.

Table continues on the next page...

I2C_IBAD field descriptions (continued)

Field	Description
	NOTE: The default mode of I^2C Bus is slave mode for an address match on the bus.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.4.4 I2C Bus Frequency Divider Register (I2C_IBFD)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 1h offset = 1h

Bit	0	1	2	3	4	5	6	7
Read Write	IBC							
Reset	0	0	0	0	0	0	0	0

I2C_IBFD field descriptions

Field	Description
0–7 IBC	I-Bus Clock Rate. This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows: 7-6 Selects the prescaled shift register (see Clock rate and IBFD settings) 5-3 Selects the prescaler divider (see Clock rate and IBFD settings) 2-0 Selects the shift register tap point (see Clock rate and IBFD settings)

55.4.5 I2C Bus Control Register (I2C_IBCR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	0
Write						RSTA		
Reset	1	0	0	0	0	0	0	0

I2C_IBCR field descriptions

Field	Description
0 MDIS	Module disable. This bit controls the software reset of the entire I^2C Bus module. **NOTE:** If the I^2C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I^2C Bus module losing arbitration, after which, bus operation would return to normal. 0 The I^2C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled.
1 IBIE	I-Bus Interrupt Enable. 0 Interrupts from the I^2C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I^2C Bus module are enabled. An I^2C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. This field is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode
3 TXRX	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
4 NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I^2C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. **NOTE:** Values written to this bit are only used when the I^2C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
5 RSTA	Repeat Start. Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect 1 Generate repeat start cycle
6 DMAEN	DMA Enable. When this bit is set, the DMA Tx and Rx lines will be asserted when the I^2C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I^2C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See the DMA Application Information section for more details.

Table continues on the next page...

I2C_IBCR field descriptions (continued)

Field	Description
	0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.4.6 I2C Bus Status Register (I2C_IBSR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 3h offset = 3h

Bit	0	1	2	3	4	5	6	7
Read	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
Write				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

I2C_IBSR field descriptions

Field	Description
0 TCF	Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. **NOTE:** This bit is only valid during or immediately following a transfer to the I^2C module or from the I^2C module. 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. 0 Bus is Idle 1 Bus is busy
3 IBAL	Arbitration Lost.

Table continues on the next page...

I2C_IBSR field descriptions (continued)

Field	Description
	The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: • SDA is sampled low when the master drives a high during an address or data transmit cycle. • SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SRW	Slave Read/Write. When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
6 IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: • Arbitration lost (IBAL bit set) • Byte transfer complete (TCF bit set and DMAEN bit not set) • Addressed as slave (IAAS bit set) • NoAck from Slave (MS & Tx bits set) • I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

55.4.7 I2C Bus Data I/O Register (I2C_IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

NOTE

The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the

transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required R/ $\overline{W}$ bit (in position D0).

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

I2C_IBDR field descriptions

Field	Description
0–7 DATA	Data transmitted or received

55.4.8 I2C Bus Interrupt Config Register (I2C_IBIC)

To program BIIE = 1, you must ensure that IBCR[MDIS] = 0.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 5h offset = 5h

Bit	0	1	2	3	4	5	6	7
Read Write	BIIE	BYTERXIE	0					
Reset	0	0	0	0	0	0	0	0

I2C_IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I^2C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I^2C bus.

Table continues on the next page...

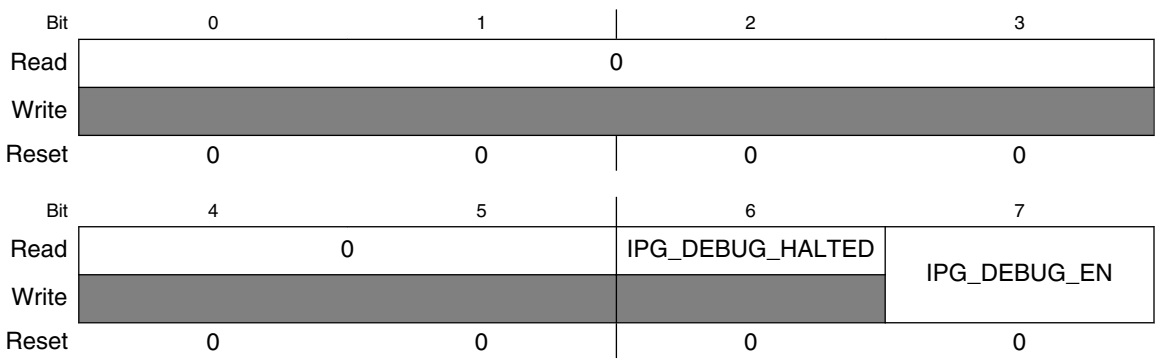
I2C_IBIC field descriptions (continued)

Field	Description
	0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1 BYTERXIE	Byte receive interrupt enable This field is used to generate an interrupt every time the I^2C master/slave receives a new byte. This feature can be useful when an I^2C master is receiving data from a slave that is transmitting on an irregular basis.
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.4.9 I2C Bus Debug Register (I2C_IBDBG)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 6h offset = 6h



I2C_IBDBG field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IPG_DEBUG_ HALTED	Debug Halted Bit: This is s status bit which can be read back after asserting the debug enable signal so as to know if the IP has entered the debug mode or not. 0 IP is still executing a transaction 1 IP has entered the debug mode
7 IPG_DEBUG_EN	Debug enable bit. This bit is used to enable IP enter the debug mode provided the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. **NOTE:** If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited.

Table continues on the next page...

I2C_IBDBG field descriptions (continued)

Field	Description
	0 Normal operation, Bus Idle Interrupts disabled 1 IP is in debug mode

55.5 Functional description

This section presents the following topics:

- Notes about module operation
- Transactions
- Arbitration procedure
- Clock behavior
- Interrupts
- IPG DEBUG mode
- DMA interface

55.5.1 Notes about module operation

- The I^2C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I^2C module is acting as a master, it must not try to call its own slave address.

55.5.2 Transactions

This section presents the following topics:

- Protocol overview
- Transaction protocol definitions
- High-level protocol steps
- START condition

- Slave address transmission
- Data transmission
- STOP condition
- Repeated START condition

55.5.2.1 Protocol overview

The following figure shows the behavior of SCL and SDA during a typical I²C transaction.

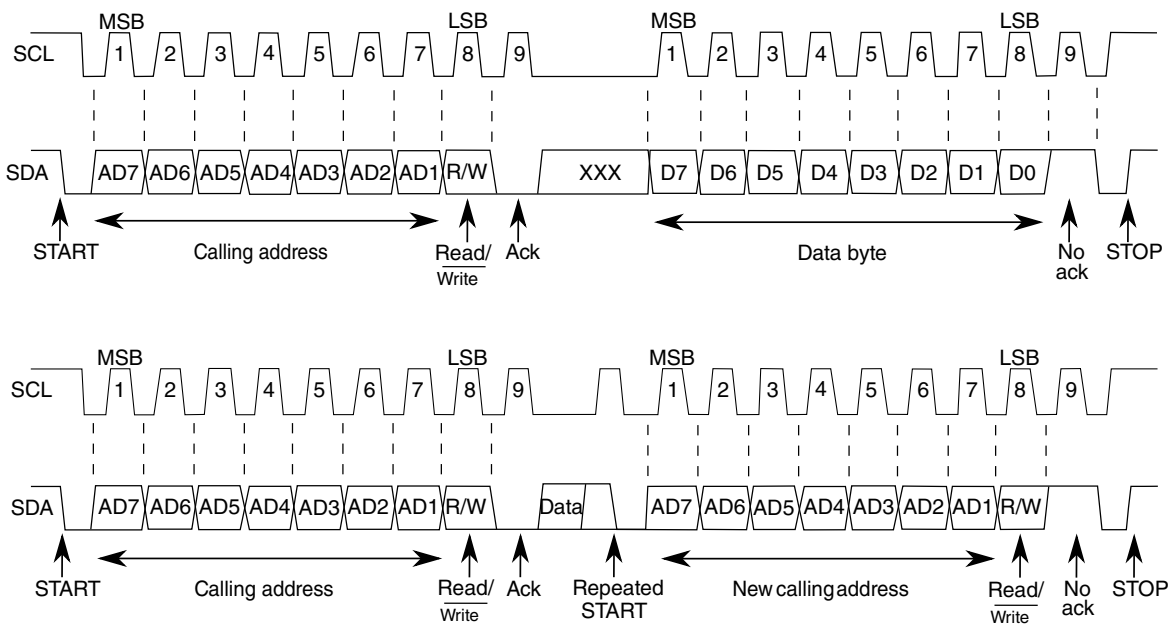


Figure 55-4. I²C transaction protocol

55.5.2.2 Transaction protocol definitions

This section defines several important terms presented in Figure 55-4.

Table 55-5. I²C definitions

Term	Definition
START	A START condition, as defined in Section 1.2.6, Definition: I²C conditions"
STOP	A STOP condition, as defined in Section 1.2.6, Definition: I²C conditions"
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in Section 1.5.2.3, I²C calling address requirements."

Table continues on the next page...

Table 55-5. I²C definitions (continued)

Term	Definition
Read/write (R/W)	A bit that specifies the direction of the data transfer to the slave as follows: • 0=The data is being transferred from the master to the slave ("write") • 1=The data is being transferred from the slave to the master ("read")
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

55.5.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

55.5.2.4 High-level protocol steps

The I²C protocol conceptually supports two types of transfers, which are illustrated in Figure 55-4. The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 55-6. I²C high-level protocol steps

Standard Transfer	Repeated START Transfer
1. START condition	1. START condition
2. Slave target or general call address transmission	2. Slave target or general call address transmission
3. Acknowledgment from slave	3. Acknowledgment from slave
4. Data transfer	4. Data transfer
5. STOP condition	5. Repeated START condition
6. (repeat Steps 1–4)	6. (repeat Steps 2–4 as needed)
	7. STOP condition.
	8. (repeat Steps 1–7)

55.5.2.5 START condition

When the bus is free, that is, no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (see Definition: I^{2}C conditions). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

55.5.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (see START condition). The process of slave address transmission is presented in the following table.

Table 55-7. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: • The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

55.5.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.

- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

55.5.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (see Definition: I^2C conditions). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, see Repeated START condition.

55.5.2.9 Repeated START condition

The I^2C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of Figure 55-4.

55.5.3 Arbitration procedure

The I^2C bus is a true multi-master bus that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters.

The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic "1" while another master transmits logic "0". The losing masters immediately switch over to slave mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

55.5.4 Clock behavior

This section presents the following topics:

- Clock synchronization
- Clock stretching
- Handshaking
- Clock rate and IBFD settings

55.5.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached.

However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see the following figure). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

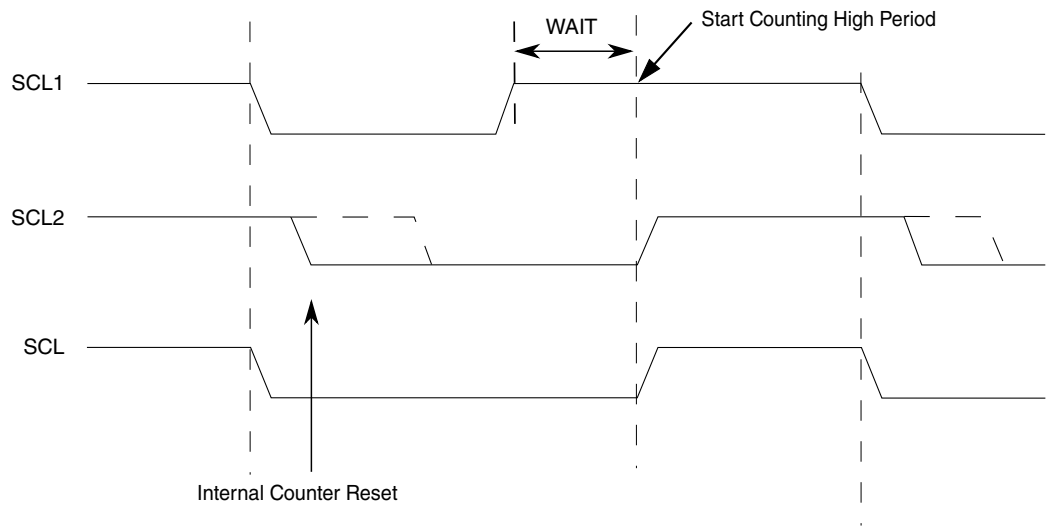


Figure 55-5. I²C bus clock synchronization

55.5.4.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

55.5.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

55.5.4.4 Clock rate and IBFD settings

The following tables describe the settings of several fields in the IBFD register (see I2C Bus Frequency Divider Register (I2C_IBFD)).

Table 55-8. I-Bus multiplier factor

IBC[0-1]	MUL
00	01
01	02
10	04

Table continues on the next page...

Table 55-8. I-Bus multiplier factor (continued)

IBC[0-1]	MUL
11	Reserved

Table 55-9. I-Bus prescaler divider values

IBC[2-4]	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 55-10. I-Bus tap and prescale values

IBC[5-7]	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 55-9. All subsequent tap points are separated by 2^{IBC} as shown in the tap2tap column in Table 55-9. The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA that is the SDA Hold time.

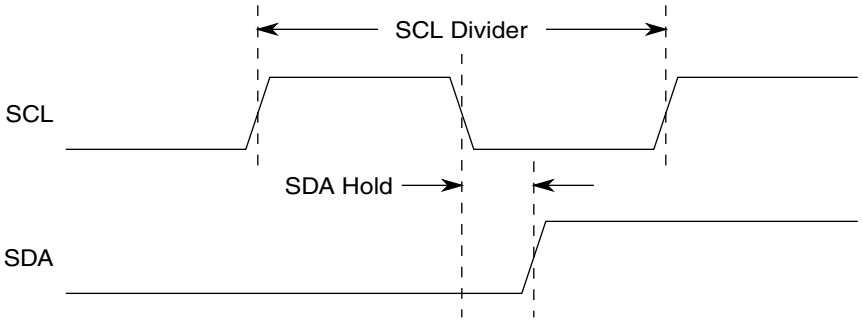


Figure 55-6. SDA Hold time

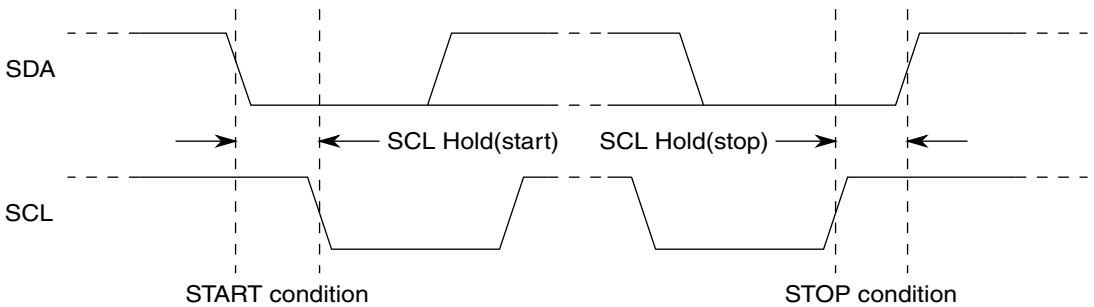


Figure 55-7. SCL Divider and SDA Hold

The equation used to generate the divider values from the IBFD bits is:

$$\text{SCL Divider} = \text{MUL} \times \{2 \times (\text{scl2tap} + [(\text{SCL\_Tap} - 1) \times \text{tap2tap}] + 2)\}$$

Equation 68. Equation for SCL Divider

The SDA Hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 55-11. The equation used to generate the SDA Hold value from the IBFD bits is:

$$\text{SDA Hold} = \text{MUL} \times \{\text{scl2tap} + [(\text{SDA\_Tap} - 1) \times \text{tap2tap}] + 3\}$$

Equation 69. Equation for SDA Hold

The equations for SCL Hold values to generate the START and STOP conditions from the IBFD bits are:

$$\text{SCL Hold (start)} = \text{MUL} \times [\text{scl2start} + (\text{SCL\_Tap} - 1) \times \text{tap2tap}]$$

Equation 70. Equation for SCL Hold(start)

$$\text{SCL Hold (end)} = \text{MUL} \times [\text{scl2end} + (\text{SCL\_Tap} - 1) \times \text{tap2tap}]$$

Equation 71. Equation for SCL Hold(stop)

Table 55-11. I²C divider and hold values

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
MUL=1	24	288	49	142	145
	25	320	49	158	161

Table continues on the next page...

Table 55-11. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921
MUL=2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
MUL=2	4A	72	18	28	38
	4B	80	18	32	42

Table continues on the next page...

Table 55-11. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	34	156	162
	61	384	34	188	194
	62	448	66	220	226
	63	512	66	252	258
	64	576	98	284	290
	65	640	98	316	322
	66	768	130	380	386
	67	960	130	476	482
	68	640	66	316	322
	69	768	66	380	386
	6A	896	130	444	450
	6B	1024	130	508	514
MUL=2	6C	1152	194	572	578
	6D	1280	194	636	642
	6E	1536	258	764	770
	6F	1920	258	956	962
	70	1280	130	636	642
	71	1536	130	764	770

Table continues on the next page...

Table 55-11. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	72	1792	258	892	898
	73	2048	258	1020	1026
	74	2304	386	1148	1154
	75	2560	386	1276	1282
	76	3072	514	1532	1538
	77	3840	514	1916	1922
	78	2560	258	1276	1282
	79	3072	258	1532	1538
	7A	3584	514	1788	1794
	7B	4096	514	2044	2050
	7C	4608	770	2300	2306
	7D	5120	770	2556	2562
	7E	6144	1026	3068	3074
	7F	7680	1026	3836	3842
MUL=4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
MUL=4	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260

Table continues on the next page...

Table 55-11. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	30	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564
MUL=4	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612
	BD	10240	1540	5112	5124

Table continues on the next page...

Table 55-11. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

55.5.5 Interrupts

This section presents the following topics:

- Interrupt vector
- Interrupt description

55.5.5.1 Interrupt vector

The I²C module uses only one interrupt vector.

Table 55-12. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I²C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set, an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

55.5.5.2 Interrupt description

There are five types of internal interrupts in the I²C. The interrupt service routine can determine the interrupt type by reading the Status register.

I²C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)

- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I2C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIE] bit.

55.5.6 IPG STOP mode

This mode allows the software to put the I2C module in power-down state. Once the STOP request is asserted, the I2C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I2C module enters IPG STOP mode:

- The I2C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I2C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. The user must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission the user must wait until the transmission is complete, followed by clearing of the IBCR[TXRX] field. See the figure below for more details.

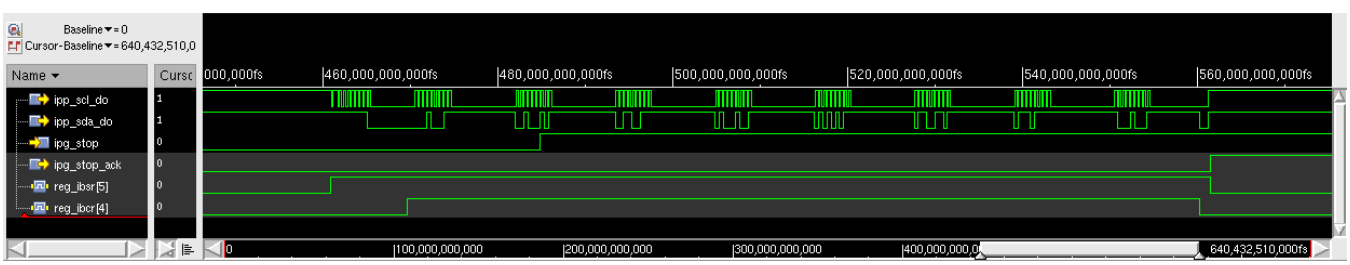


Figure 55-8. I2C stop mode behavior when master is receiving and slave is transmitting

55.5.7 IPG DEBUG mode

This mode allows CPU to debug the I2C by freezing all the counters, registers and status bits. Once the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] bit, I2C comes to a graceful halt after completing all the ongoing transactions. A Debug halted

signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I^2C module enters IPG DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

The I^2C module enters this mode only after successfully completing the current ongoing transaction. For example, if the current transaction consists of 8 bytes and the debug mode was initiated by user at the time of the second byte, the IPG Debug Halted signal will be asserted after the 8th byte is transmitted/received. See the simulation result in Figure 55-9 for more details.

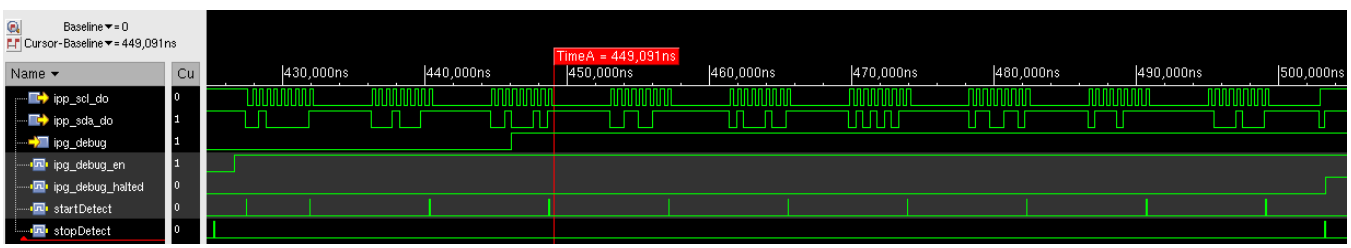


Figure 55-9. Simulation result of IPG Debug Halted in case of frame transmission

If any DMA transaction (transmit or receive) is in progress, and if the debug mode is requested while a byte is being transmitted or received, the I^2C module enters IPG DEBUG mode after completing the transmission or reception of the current byte. As soon as the module exits IPG DEBUG mode, transmission or reception of the remaining bytes resumes. Please see the simulation snapshot shown in Figure 55-10 for details where the I^2C is transmitting 8 bytes using DMA and IPG DEBUG mode is requested while a second byte is being transmitted. IPG DEBUG mode is entered after successfully transmitting the second byte (IPG Debug Halted signal asserted). As soon as the module exits IPG DEBUG mode (IPG Debug Halted signal de-asserted), transmission resumes successfully.

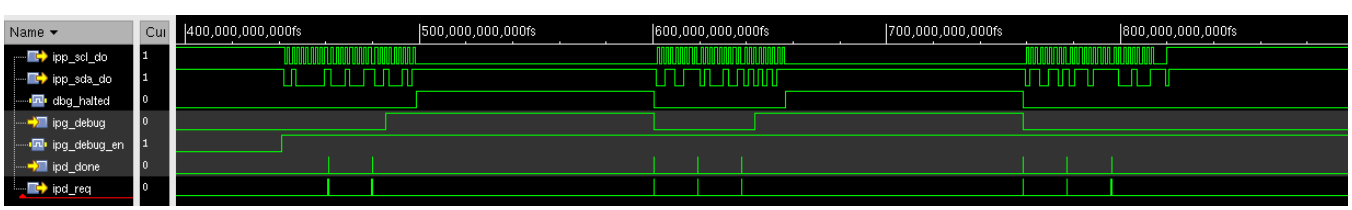


Figure 55-10. Simulation result of IPG Debug Halted in case of frame transmission using DMA

No more transaction can take place until the debug signal is de-asserted after which the I^2C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I^2C module has entered the IPG DEBUG mode. See the following figure for more details.

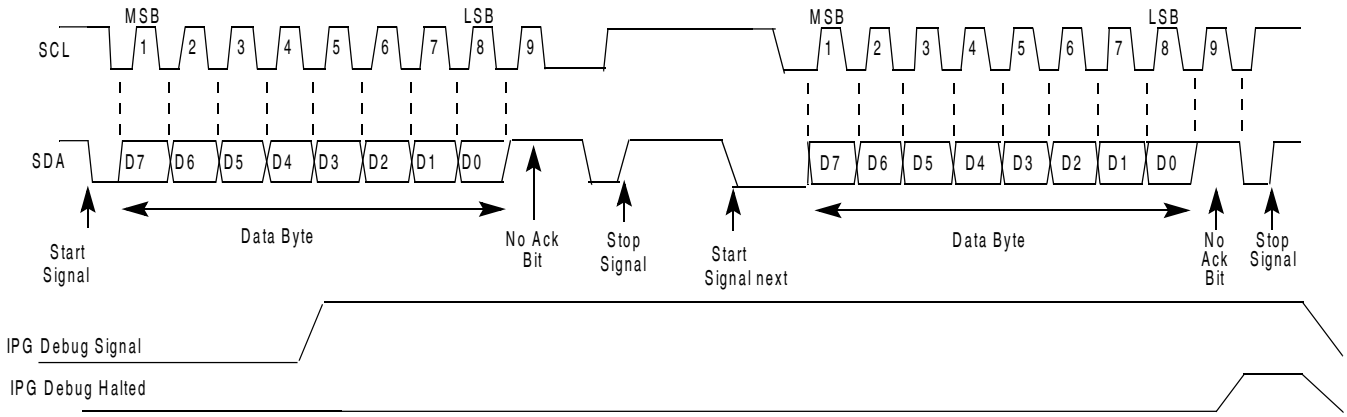


Figure 55-11. IPG DEBUG mode

The following figure shows a case of IPG DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I^2C module enters IPG DEBUG mode.

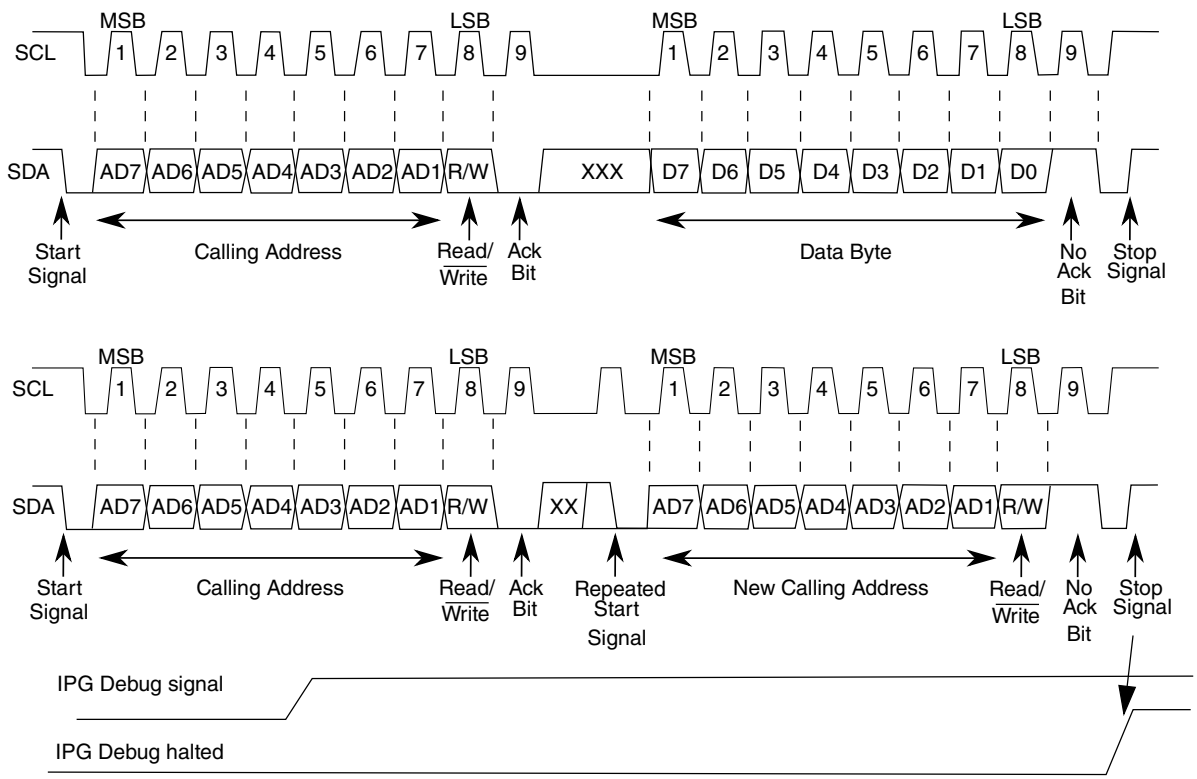


Figure 55-12. IPG debug mode with repeated start

55.5.8 DMA interface

A simple DMA interface is implemented so that the I^2C can request data transfers with minimal support from the CPU (see DMA application information). DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I^2C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I^2C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the Initialization/application information, of this document.

55.6 Initialization/application information

This section presents the following topics:

- Recommended interrupt service flow
- General programming guidelines (for both master and slave mode)
- Programming guidelines specific to master mode
- Programming guidelines specific to slave mode
- DMA application information

55.6.1 Recommended interrupt service flow

The following figure shows a flowchart for the recommended I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior.

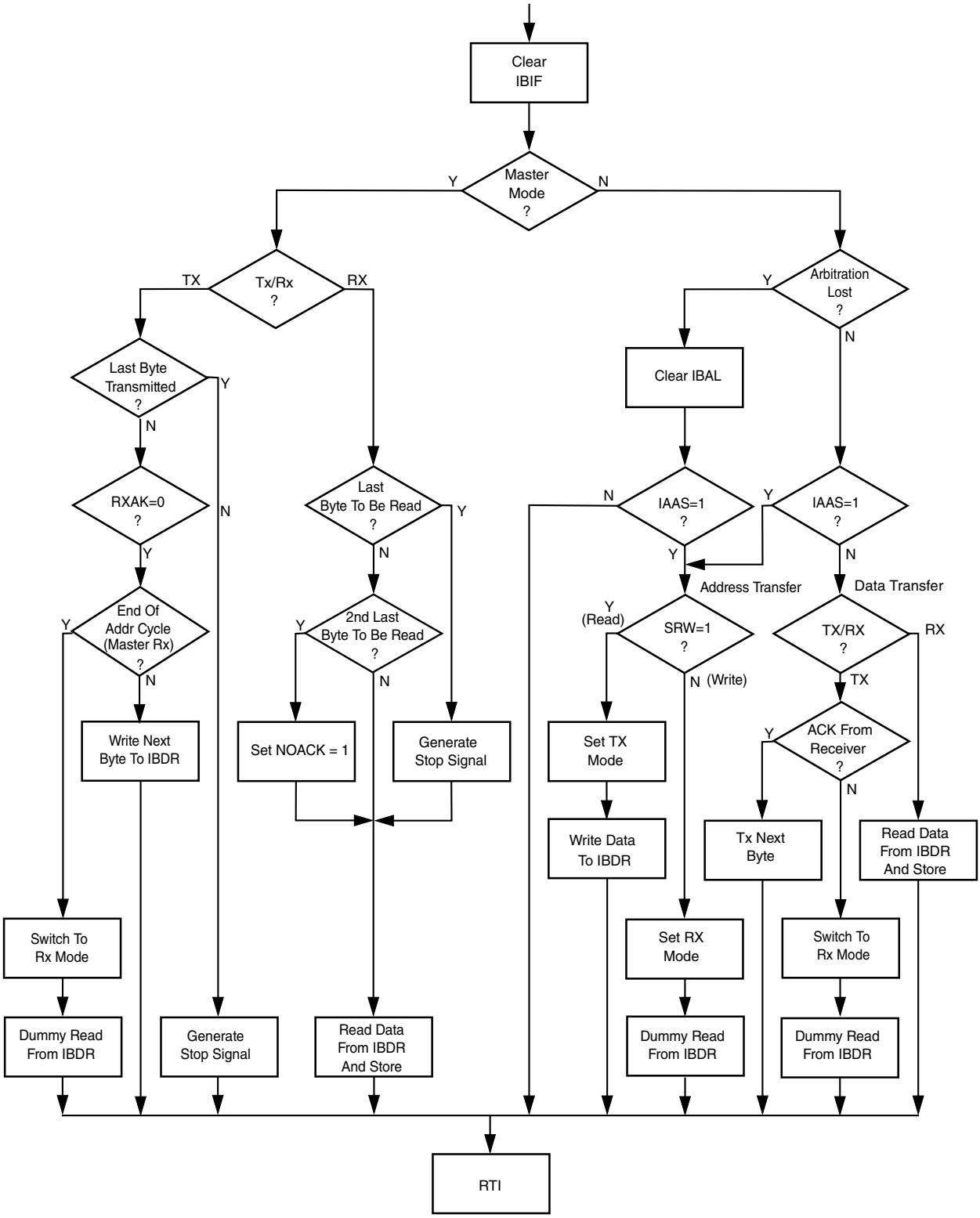


Figure 55-13. Recommended I²C interrupt service routine flowchart

55.6.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I^2C module in both master and slave mode. It presents the following topics:

- Section 1.6.2.1, Initializing the I^2C module
- Software response after a transfer

55.6.2.1 Initializing the I^2C module

The following sequence initializes the I^2C module:

1. Use the IBFD register to select the required division ratio to obtain SCL frequency from system clock.
2. Use the IBAD register to define the slave address.
3. Clear the IBCR[MDIS] bit to enable the I^2C interface system.
4. Use the IBCR to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not.
5. (Optional) Use the IBIC register to further refine the interrupt behavior.

55.6.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I^2C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I^2C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^2C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

When a "Transfer Complete" interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

55.6.3 Programming guidelines specific to master mode

This section presents the following topics:

- Generating START
- Transmit/receive sequence
- Generating STOP
- Generating repeated START
- Loss of arbitration

55.6.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)                // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1  // master and transmit mode, that is,
                                    // generate start condition
IBDR = calling_address              // send the calling address to the data register
while (bit 5, IBSR ==0)             // wait in loop for IBB flag to be set
```

55.6.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 55-13. Master transmit sequence

Step	Action
a	Use the IBFD register to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to IBCR[IBDIS] to enable the I^2C interface system.
c	Use the IBCR to select Master mode, Transmit mode and interrupt enable.
d	Write 0 to IBSR[IBIF].
e	Write data to IBDR.
f	Observe changes in IBSR[TCF]: • When IBSR[TCF] becomes 0, the transfer is in progress. • When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until IBSR[IBIF] = 1.
h	Read the fields in the IBSR to determine what happened: • If TCF = 1, the transfer completed. • If RXAK = 1, a No Acknowledge condition occurred. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore Address Detect (IAAS = 1) for master mode (it is valid only for slave mode).
i	Examine IBSR[RXAK] for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 55-14. Master receive sequence

Step	Action
a	Follow steps a through i in Table 55-13 for address dispatch.
b	Write 0 to IBSR[IBIF].
c	Write 0 to IBCR[TXRX] to select Receive mode.
d	Perform a dummy read of the IBDR to initiate the receive operation.
e	Wait until IBSR[TCF] becomes 1. (This proves that the transfer is complete.)
f	Wait until IBSR[IBIF] = 1.
g	Read the fields in the IBSR to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read the IBDR to determine the data received from the slave.

Table 55-15. Slave transmit sequence

Step	Action
a	Use the IBAD register to define the slave address.
b	Write 0 to IBCR[IBDIS] to enable the I^2C interface system.
c	Examine fields in the IBSR as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.
d	Write data to IBDR.
e	Wait until IBSR[IBIF] = 1.
f	Wait until IBSR[RXAK] = 0.
g	Write 0 to IBSR[IBIF].
h	Repeat steps d through g for the next consecutive data transfers.

Table 55-16. Slave receive sequence

Step	Action
a	Use the IBAD register to define the slave address.
b	Write 0 to IBCR[IBDIS] to enable the I^2C interface system.
c	Examine fields in the IBSR as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
d	Write 0 to IBSR[IBIF].
e	Perform a dummy read of the IBDR to initiate the receive operation.
f	Wait until IBSR[TCF] becomes 1. (This proves that the transfer is complete.)
g	Wait until IBSR[IBIF] = 1.
h	Read the fields in the IBSR to determine what happened: • If TCF = 1, the reception completed.

Table continues on the next page...

Table 55-16. Slave receive sequence (continued)

Step	Action
	• If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
i	Read the IBDR to determine the data received from the master.

55.6.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```
if (tx_count == 0) or     // check to see if all data bytes have been transmitted
  (bit 0, IBSR == 1) {    // or if no ACK generated
    clear bit 5, IBCR     // generate stop condition
  }
else {
 IBDR = data_to_transmit  // write byte of data to DATA register
   tx_count --            // decrement counter
 }                        // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --             // decrease the rx counter
if (rx_count ==1)       // 2nd last byte to be read ?
 bit 3, IBCR = 1        // disable ACK
 if (rx_count == 0)     // last byte to be read ?
 bit 5, IBCR = 0        // generate stop signal
else
data_received = IBDR    // read RX data and store
```

55.6.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1             // generate another start (restart)
IBDR == calling_address     // transmit the calling address
```

55.6.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU, set the IBAL to indicate that the attempt to engage the bus is failed, and not set the TCF due to the loss of data during arbitration. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

55.6.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an "end of data" signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

55.6.5 DMA application information

The DMA interface on the I^2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMA enable bit in the control register is not set when the I^2C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The DMAEN bit in the IBCR register works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, dependant on the setting of the DMAEN bit. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place. The sections present the following topics:

- DMA mode, master transmit
- DMA mode, master reception
- Exiting DMA mode, system requirement considerations

55.6.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit "n" data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

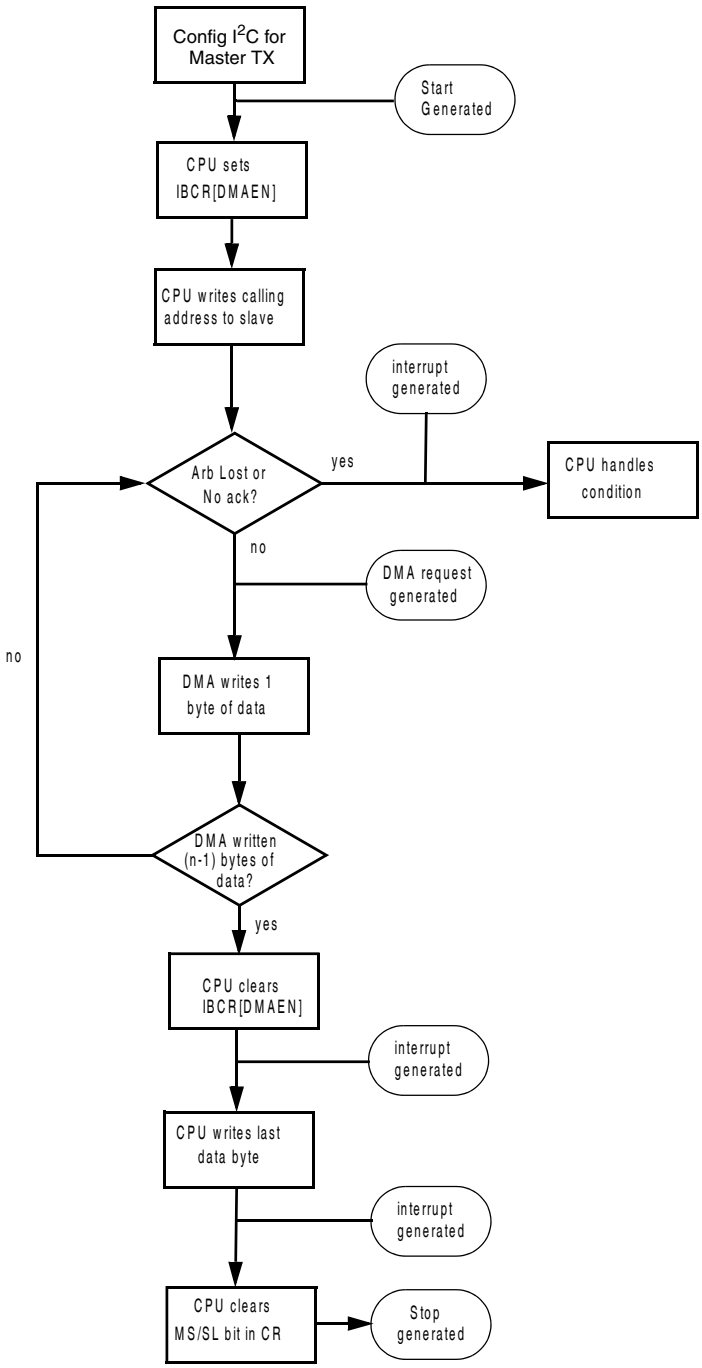


Figure 55-14. Flow-Chart of DMA mode master transmit

55.6.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive "n" data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

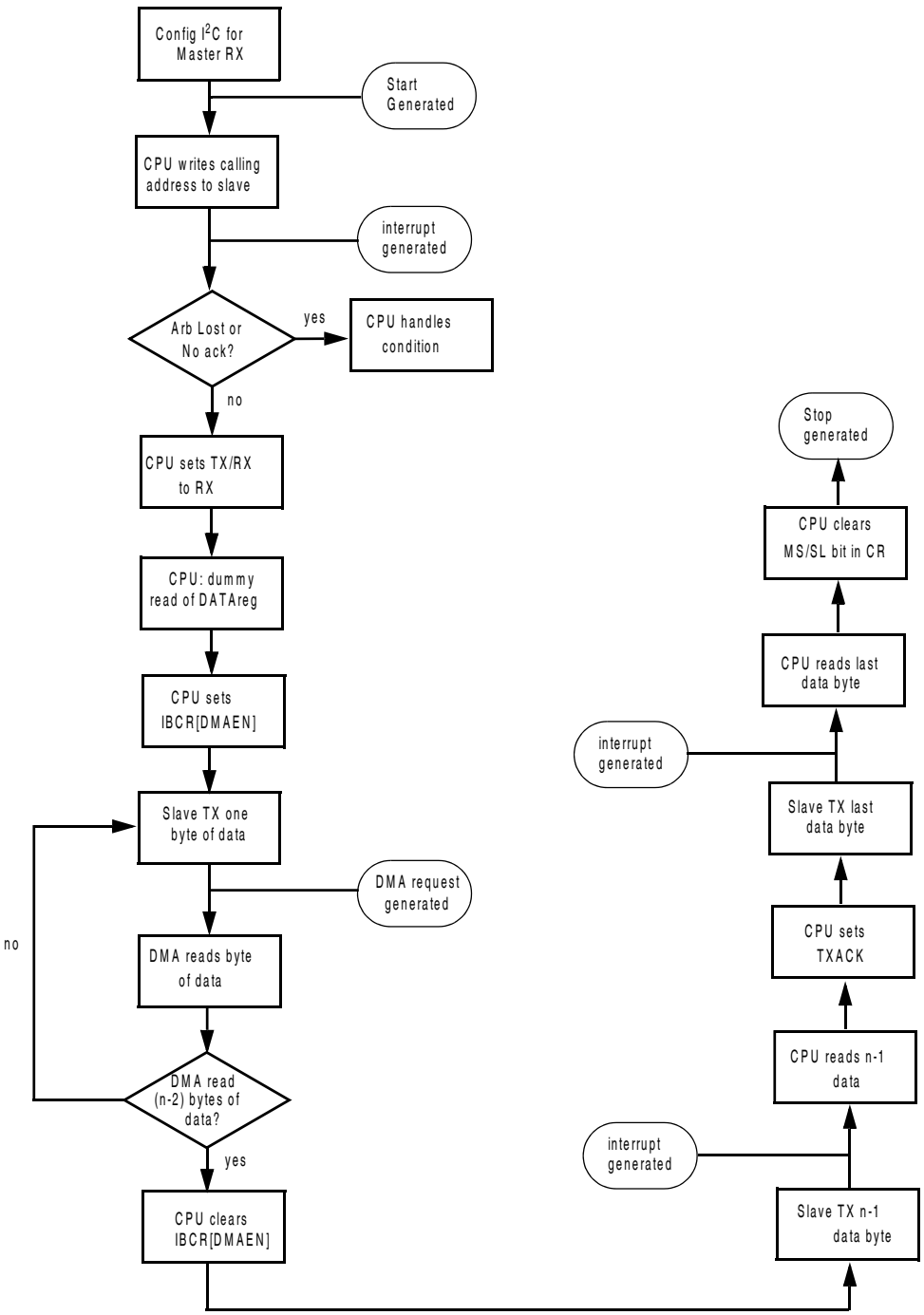


Figure 55-15. Flow-Chart of DMA mode master receive

55.6.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I^2C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I^2C module.

After the last DMA write (TX mode) to the I^2C the module will immediately start the next I^2C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I^2C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

   The DMAEN bit is cleared before the next I^2C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

   The DMAEN bit is cleared after the next I^2C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I^2C is the time required for one byte transfer over the I^2C. For 'fast reaction' the disabling has to occur before the 9^{th} bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

$$T_R = 8 \times T_{SCL}$$

In fast mode, with 400kbit/s, T_{SCL} is 2.5µs, so T_R is 20µs.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

   The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority

interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act accordingly. In case of slow reaction you can force an interrupt for the I^2C in the interrupt controller to have the further transfer handled by the normal I^2C interrupt routine.

Note

The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking

   If the Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to IBCR to disable the DMAEN field, this might probably be the fastest system solution, but it uses two DMA channels.

   Note

   Here you have to make sure on system level that no higher priority DMA requests occur between the two linked TCDs as those could again create a scenario of slow reaction.

3. DMA scatter/gather process

   If the Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated, this feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the IBCR to disable the DMAEN field. This TCD also has scatter/gather activated and is programmed to reload the initial TCD upon completion, bringing the system back into a "ready-for-I^2C-transfer" state. The advantage over the two other solutions is that this requires neither CPU intervention nor a second DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note

Here you have to make sure at system level that no higher priority DMA requests occur during the scatter/gather process, as those could again create a scenario of slow reaction.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/16 MHz = 250 ns

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

  $8 \times T_{AHD}$ = 8/32 MHz = 250 ns

Example latencies for a 150 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/150 MHz = 26.6 ns

- Reloading a new TCD (4 x 64-bit) via AHB to the DMA controller (scatter/gather process):

  $4 \times T_{AHD}$ = 4/150 MHz = 26.6 ns

With the DMA scatter/gather process the required IBCR access can be done in 0.5 µs, leaving a large margin of 19.5 µs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

Chapter 56
Peripheral Sensor Interface (PSI5)

56.1 Introduction

The PSI5 interface offers power supply and data transfer between the application and multiple sensors over a two-wire interface. Data transmission from the application to the sensor or sensors is achieved through modulation of the power supply voltage (sync pulse). Data reception from the sensor or sensors to the application is achieved through modulation of the current drawn by a sensor using Manchester encoding.

At the system level, the PSI5 interface consists of mechanisms integrated on the microcontroller and an external physical layer that provides the high-voltage supply and converts the current drawn by the sensor into a digital Manchester-encoded data stream.

56.1.1 Overview

The PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make the PSI5 also suitable for many other automotive sensor applications. The development goal of the PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented free of charge.

The main features of the PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications. The main features of PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 Kbit/s

- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

In short, PSI5 standardizes the low-level communication between peripheral sensors and electronic control units. Note that the physical layer for the IP (transreceiver) is to be implemented separately outside the IP.

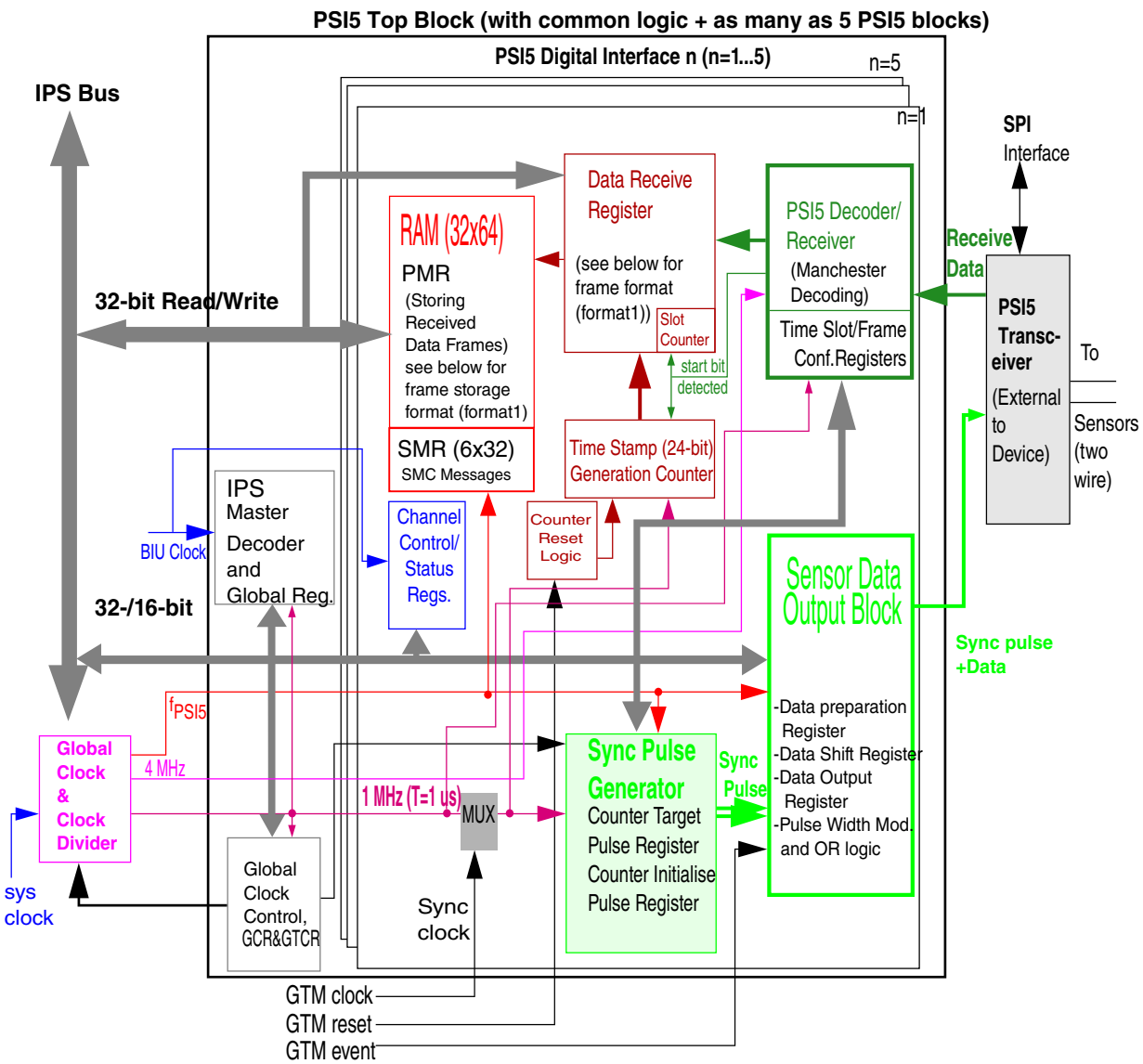


Figure 56-1. PSI5 digital interface block diagram

56.1.2 Features

- IPS interface for register read/write (big endian)
- Supports bidirectional full duplex operations, but with differing baud rate—low, ECU-to-sensor, fast sensor to ECU
- Manchester decoder operating at 125 Kbit/s bit rate with ×32 sample clock rate
- One 64-bit sensor data receive register to which the decoded received sensor data—including header, data, timestamp, frame counter, status bits (E, EM, T, C, F) and CRC, but not including start bits—are written
- A 24-bit time stamp register with a 1 μs (or the GTM clock) resolution. When first rising edge of start bits (S0) of a message or the rising edge of the SYNC pulse is recognized, the value in the time stamp register will be written to the data receive register
- Provision of resetting the Time Stamp Counter through the reset from the GTM module
- Single 32 × 64-bit RAM buffer for storing (moving) content of the data receive register once end of message is recognized
    - Related to each RAM buffer are two diagnostic registers, N and EI. Register N is used to identify the occurence of new messages in the corresponding RAM buffer, while register EI identifies the error status of these messages. The errors that can set the EI register bits are the F (no frame), E (electrical error), EM (electrical error for SMC bits embeddded in PSI5 messages), T (Timing bit error) and C (CRC error). Which of these errors actually set the EI register bits is configurable.
- DMA support for transferring RAM data (including diagnostic data) to system memory
- Support for optional serial messaging channel (SMC) and additional logic to extract SMC frames from PSI5 frames
- Operating modes supported:
    - Asynchronous mode (sensor periodically sends data, no sync pulse, one sensor per interface)
    - Synchronous mode (sync pulse generated, multiple sensor per interface)
- Sync pulse generation for synchronous modes

- The start of each time slot (1 to 6) (see Figure 56-6) on the bus (after the end of the sync pulse) can be defined with a 1 µs resolution
- The end of the last time slot on the bus can be defined with a 1 µs resolution, up to a maximum sync period time of 16 ms
- Trigger logic to generate sync pulses
  - 16-bit Channel Trigger Counter (CTC) used to generate the sync pulses by comparing the values with CIPR and CTPR
  - Counter Initialize Pulse Register (CIPR) to allow configuration of initialization (offset) period
  - Channel Trigger Period (CTP) register to allow configuration of sync period. Value in CTP is the value to which the CTC is updated after each sync pulse generation
  - Possibility to have a common 1 MHz clock source

- An upstream register bit chain consisting of Preparation Register, Buffer Register, Shift register (output data) and pulse width modulation logic, is implemented. Using this, it is possible to send a data word to the sensors through use of data sync pulses: logic 1 with programmable data sync pulse width W1, logic 0 with programmable data sync pulse width W0. The data sync pulse timing can be configured to be dependent on the periodic or event driven timing SYNC pulse. Further, there is no difference between timing sync pulse and data sync pulse during transmission (ECU-to-sensor communication).

56.1.3 Modes of operation

56.1.3.1 Disable mode

Disable mode in the PSI5 reduces power consumption. This mode is entered by setting GCR[GLOBAL_DISABLE_REQ]. All the channels enter Disable mode at once when this bit is set, irrespective of the setting of PCCR[PSI5_CH_EN] and PCCR[PSI5_CH_CONFIG]. In this mode the RX, TX and common time base counters are disabled for any operation. This is the default state after module reset is released. PSI5 can exit Disable mode by clearing the GLOBAL_DISABLE_REQ bit and enabling individual PSI5 channels (setting PCCR[PSI5_CH_EN]). Note that when entering

Disable mode from any other mode, the register contents are retained and are not changed. The time stamp counter AND the internal sync pulse generator are, however, reset when entering Disable mode.

There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter "Configuration/Normal mode" based on the value of PCCR[PSI5_CH_CONFIG].

56.1.3.2 Initialization/Configuration mode

This is the configuration phase, in which, all PSI5 global, channel specific configuration/ control registers can be written. The receiver and transmitter sub-blocks are disabled in this mode. The message receive registers/buffers can be accessed in this mode. This mode can be entered by clearing GCR[GLOBAL_DISABLE_REQ] and setting PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN].

56.1.3.3 Normal mode (NORMAL mode)

In Normal mode, the reciever and the transmitter blocks become active, error handling is enabled and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5 global and channel-specific configuration registers are locked for writing, although these registers can still be read. Data Preparation Register, Shift Register and data output registers are writable in this mode. Apart from them, there are some register bits that can be written in this mode also. Please refer to the detailed register description. This mode can be entered clearing GCR[GLOBAL_DISABLE_REQ], clearing PCCR[PSI5_CH_CONFIG], and setting PCCR[PSI5_CH_EN].

56.1.3.3.1 PSI5 topology

The different PSI5 user operation modes define topology and parameters of the communication between ECU and sensors such as communication mode, number of data bits, error detection, cycle time, number of time slots per cycle, and bit rate.

These modes as generally denoted as

PSI5-(A / P / U / D)-dd-(P / CRC)-tttt / n(L / H)

The following figure provides a key for these notations.

Table 56-1. Normal operating states

Communication modes		Description
A	Asynchronous mode	PSI5-A describes a point-to-point connection for unidirectional, asynchronous data transmission. Each sensor is connected to the ECU by two wires. After switching on the power supply, the sensor starts transmitting data to the ECU periodically. Timing and repetition rate of the data transmission are controlled by the sensor. No Sync pulse is send from ECU side.
P	Synchronous Parallel Bus mode	The synchronous operation modes work according to the TDMA method (Time Division Multiple Access). Each sensor is connected to the ECU by two wires. The sensor data transmission is synchronized by the ECU using voltage modulation. Synchronization (sync pulses) can be optionally used for point-to-point configurations; it is mandatory for bus modes. The sensors can be arranged as parallel bus, universal bus and daisy chain topology based on sensor addressing and interchangeability of sensors.
U	Synchronous Universal Bus mode	
D	Synchronous Daisy Chain Bus mode	
Data Bits		
dd	No. of data bits	
Error Detection		
P	1 parity bit	
CRC	3-bit Cyclic Redundancy Check (CRC)	
Cycle Time		
tttt	Cycle Time in µs	
n	Number of time slots per cycle	
Bit Rate		
L	125 Kbit/s	

56.1.3.4 Stop mode

When the Stop mode request is asserted, and if there is no active PSI5 transaction ongoing, then the acknowledgement to stop the clocks is given immediately .If start bits of any PSI5 packet have been detected(sucessful detection of both S0 and S1) then the concerned PSI5 packet is sucessfully stored and then the Stop mode acknowledgement is generated by the IP. Note that for generating the acknowledgement, the IP does not wait for the complete reception of any ongoing SMC message, but only waits for the complete reception of any ongoing PSI5 packet transaction.

If the PSI5 transceiver is asynchronous to the PSI5 IP receiver then loss of data can result when the STOP mode is deasserted. Hence, the following guidelines should be followed:

- If the IP is in synchronous mode of operation, then the IP gets resynchronized on the next available sync pulse which occurs after STOP mode de-assertion. Hence, the PSI5 packets occurring between the STOP mode de-assertion event and the subsequent synch pulse would not be reliable and should be ignored by the application.
- If the IP is configured in asynchronous mode, then it should be ensured that the idle time gap between PSI5 packets is greater than or equal to the length of one PSI5 message. In this case, only one PSI5 message will get corrupted, but the IP will sample the subsequent PSI5 messages correctly.

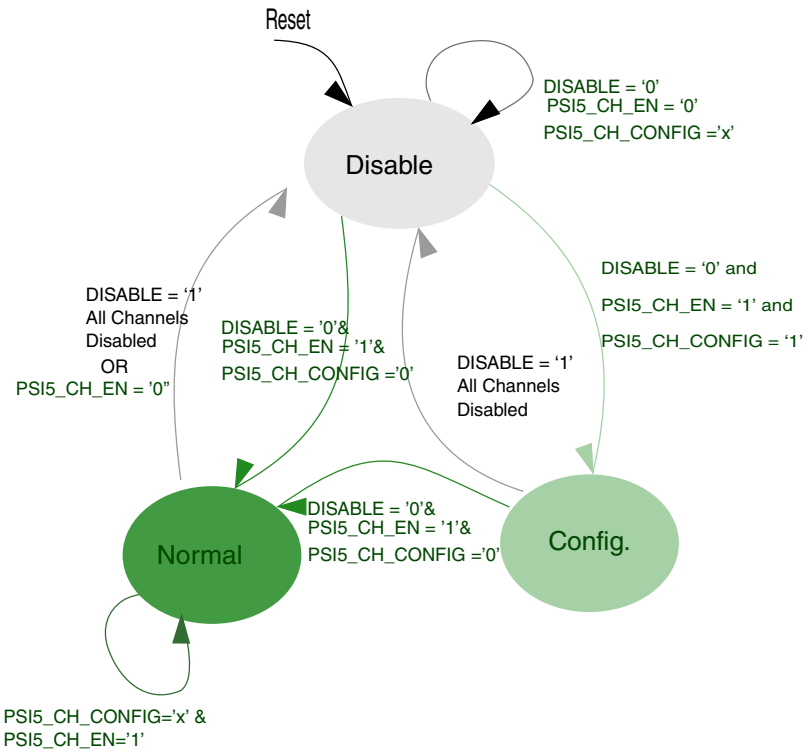
56.1.3.5 Debug mode

In order to enter this mode the debugger has to be connected and it has to encounter a breakpoint. In addition, the DEBUG_EN bit has to be written to 1. The DEBUG_EN bit can be written by either the CPU or by the debugger but only in Configuration mode. Once the above conditions are satisfied the IP enters the Debug Mode. There are two debugging modes available. For simplicity these are being referred to as the “Freeze” and “Open” modes. The IP enters one of the above debugging modes, based on the configuration of a register bit DEBUG_ FREEZE_ CTRL mentioned in the Section 54.3.2.2, PSI5 Channel Control Register (PSI5_PCCR).

In Open mode, the core of the IP keeps on working normally and all the registers are updated from the hardware; just like they were being done in the functional mode (when the IP was out of the debug mode). Thus, even a long term entry to the Debug Mode, wont affect the normal operation of the various registers that are to be updated by the hardware. Down side is that the IP can keep on changing the internal status bits and the registers as the time progresses.

In both the above modes, the various interface registers would be completely visible to the debugger, but debugger action would have no impact on the various status bits and counters that normally get affected by a CPU/DMA read. In this mode, the read pointers of the FIFO do not get incremented, when being read through the Pop-up registers. Thus in Debug mode, reading the DMA PSI5 Message Register (PSI5_DPMR),DMA SMC Frame Register (PSI5_DSFR) or the DMA Diagnostic Status Register (PSI5_DDSR) would keep on returning the same data. Hence the interrupts related to DMA completion operation would not get generated. Further the hardware clearing related to FAST_CLR _SMC and FAST_CLR _PSI5 bits would not have any effect and the hardware clearing would remain disabled. This is also highlighted in the following figure.

56.1.3.6 Operating modes



Note Not possible to reconfigure the PSI5 on the fly. A direct transition from the Normal to Config mode is not there to consider safety aspects.

Figure 56-2. Operating modes

Table 56-2. State transition Table

GCR	PCCR		State	
[GLOBAL_DISABLE_REQ]	[PSI5_CH_CONFIG]	[PSI5_CH_EN]	Present	Next
1	x	x	x	All channel Disable
0	x	0	x	Specific channel Disable
0	1	1	Disable	Config
0	0	1	Config	Normal
0	x	1	Normal	Normal
0	0	1	Disable	Normal

56.2 External signal description

External signal description.

Table 56-3. PSI5 port list

Name	Function	I/O	Reset	Pullup
Signals to/from pads				
SDINn_PSI5_m[1]	Digital sensor data input pin of PSI5 module	I	0	Active
SDOUTn_PSI5_m[1]	Digital sensor data output pin of PSI5 module (sync pulse + data)	O	0	Passive

1. “n” is the Channel Number and “m” is the Module Instance Number.

56.3 Memory map and register description

The module memory map is shown below.

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode.

Note that , an access(read/write) to any register in the unimplemented register space will result in the generation of transfer error . All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to do a word operation on 16-bit registers will NOT result in the generation of any transfer error. Trying to write to "read only" bits of implemented register space, will be ignored without the generation of any error. Trying to write to any register outside its intended mode of write(Normal/ config) will be ignored without the generation of any transfer error.

For a summary of the register bit accesses in various device modes, please see Device modes and Register bit accesses

PSI5 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Control Register (PSI5_GCR)	16	R/W	0001h	56.3.1/2758
8	PSI5 Channel Control Register (PSI5_CH0_PCCR)	32	R/W	001F_0000h	56.3.2/2759
C	DMA Control Register (PSI5_CH0_DCR)	32	R/W	0000_0000h	56.3.3/2764
10	DMA Status Register (PSI5_CH0_DSR)	32	w1c	0000_0000h	56.3.4/2768
14	General Interrupt Control Register (PSI5_CH0_GICR)	32	R/W	0000_0000h	56.3.5/2770
18	New Data Interrupt Control Register (PSI5_CH0_NDICR)	32	R/W	0000_0000h	56.3.6/2772
1C	Overwrite Interrupt Control Register (PSI5_CH0_OWICR)	32	R/W	0000_0000h	56.3.7/2773
20	Error Interrupt Control Register (PSI5_CH0_EICR)	32	R/W	0000_0000h	56.3.8/2773
24	General Interrupt Status Register (PSI5_CH0_GISR)	32	w1c	0007_0000h	56.3.9/2774

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	DMA PSI5 Message Register (PSI5_CH0_DPMR)	32	R	0000_0000h	56.3.10/ 2776
2C	DMA SMC Frame Register (PSI5_CH0_DSFR)	32	R	0000_0000h	56.3.11/ 2777
30	DMA Diagnostic Status Register (PSI5_CH0_DDSR)	32	R	0000_0000h	56.3.12/ 2777
34	PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)	32	R	0000_0000h	56.3.13/ 2778
38	PSI5 Message Receive Register High (PSI5_CH0_PMRRH)	32	R	0000_0000h	56.3.14/ 2779
3C	PSI5 Message Register Low i (PSI5_CH0_PMRL0)	32	R/W	0000_0000h	56.3.15/ 2780
40	PSI5 Message Register High i (PSI5_CH0_PMRH0)	32	R/W	0000_0000h	56.3.16/ 2781
44	PSI5 Message Register Low i (PSI5_CH0_PMRL1)	32	R/W	0000_0000h	56.3.15/ 2780
48	PSI5 Message Register High i (PSI5_CH0_PMRH1)	32	R/W	0000_0000h	56.3.16/ 2781
4C	PSI5 Message Register Low i (PSI5_CH0_PMRL2)	32	R/W	0000_0000h	56.3.15/ 2780
50	PSI5 Message Register High i (PSI5_CH0_PMRH2)	32	R/W	0000_0000h	56.3.16/ 2781
54	PSI5 Message Register Low i (PSI5_CH0_PMRL3)	32	R/W	0000_0000h	56.3.15/ 2780
58	PSI5 Message Register High i (PSI5_CH0_PMRH3)	32	R/W	0000_0000h	56.3.16/ 2781
5C	PSI5 Message Register Low i (PSI5_CH0_PMRL4)	32	R/W	0000_0000h	56.3.15/ 2780
60	PSI5 Message Register High i (PSI5_CH0_PMRH4)	32	R/W	0000_0000h	56.3.16/ 2781
64	PSI5 Message Register Low i (PSI5_CH0_PMRL5)	32	R/W	0000_0000h	56.3.15/ 2780
68	PSI5 Message Register High i (PSI5_CH0_PMRH5)	32	R/W	0000_0000h	56.3.16/ 2781
6C	PSI5 Message Register Low i (PSI5_CH0_PMRL6)	32	R/W	0000_0000h	56.3.15/ 2780
70	PSI5 Message Register High i (PSI5_CH0_PMRH6)	32	R/W	0000_0000h	56.3.16/ 2781
74	PSI5 Message Register Low i (PSI5_CH0_PMRL7)	32	R/W	0000_0000h	56.3.15/ 2780
78	PSI5 Message Register High i (PSI5_CH0_PMRH7)	32	R/W	0000_0000h	56.3.16/ 2781
7C	PSI5 Message Register Low i (PSI5_CH0_PMRL8)	32	R/W	0000_0000h	56.3.15/ 2780

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	PSI5 Message Register High i (PSI5_CH0_PMRH8)	32	R/W	0000_0000h	56.3.16/ 2781
84	PSI5 Message Register Low i (PSI5_CH0_PMRL9)	32	R/W	0000_0000h	56.3.15/ 2780
88	PSI5 Message Register High i (PSI5_CH0_PMRH9)	32	R/W	0000_0000h	56.3.16/ 2781
8C	PSI5 Message Register Low i (PSI5_CH0_PMRL10)	32	R/W	0000_0000h	56.3.15/ 2780
90	PSI5 Message Register High i (PSI5_CH0_PMRH10)	32	R/W	0000_0000h	56.3.16/ 2781
94	PSI5 Message Register Low i (PSI5_CH0_PMRL11)	32	R/W	0000_0000h	56.3.15/ 2780
98	PSI5 Message Register High i (PSI5_CH0_PMRH11)	32	R/W	0000_0000h	56.3.16/ 2781
9C	PSI5 Message Register Low i (PSI5_CH0_PMRL12)	32	R/W	0000_0000h	56.3.15/ 2780
A0	PSI5 Message Register High i (PSI5_CH0_PMRH12)	32	R/W	0000_0000h	56.3.16/ 2781
A4	PSI5 Message Register Low i (PSI5_CH0_PMRL13)	32	R/W	0000_0000h	56.3.15/ 2780
A8	PSI5 Message Register High i (PSI5_CH0_PMRH13)	32	R/W	0000_0000h	56.3.16/ 2781
AC	PSI5 Message Register Low i (PSI5_CH0_PMRL14)	32	R/W	0000_0000h	56.3.15/ 2780
B0	PSI5 Message Register High i (PSI5_CH0_PMRH14)	32	R/W	0000_0000h	56.3.16/ 2781
B4	PSI5 Message Register Low i (PSI5_CH0_PMRL15)	32	R/W	0000_0000h	56.3.15/ 2780
B8	PSI5 Message Register High i (PSI5_CH0_PMRH15)	32	R/W	0000_0000h	56.3.16/ 2781
BC	PSI5 Message Register Low i (PSI5_CH0_PMRL16)	32	R/W	0000_0000h	56.3.15/ 2780
C0	PSI5 Message Register High i (PSI5_CH0_PMRH16)	32	R/W	0000_0000h	56.3.16/ 2781
C4	PSI5 Message Register Low i (PSI5_CH0_PMRL17)	32	R/W	0000_0000h	56.3.15/ 2780
C8	PSI5 Message Register High i (PSI5_CH0_PMRH17)	32	R/W	0000_0000h	56.3.16/ 2781
CC	PSI5 Message Register Low i (PSI5_CH0_PMRL18)	32	R/W	0000_0000h	56.3.15/ 2780
D0	PSI5 Message Register High i (PSI5_CH0_PMRH18)	32	R/W	0000_0000h	56.3.16/ 2781
D4	PSI5 Message Register Low i (PSI5_CH0_PMRL19)	32	R/W	0000_0000h	56.3.15/ 2780

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D8	PSI5 Message Register High i (PSI5_CH0_PMRH19)	32	R/W	0000_0000h	56.3.16/ 2781
DC	PSI5 Message Register Low i (PSI5_CH0_PMRL20)	32	R/W	0000_0000h	56.3.15/ 2780
E0	PSI5 Message Register High i (PSI5_CH0_PMRH20)	32	R/W	0000_0000h	56.3.16/ 2781
E4	PSI5 Message Register Low i (PSI5_CH0_PMRL21)	32	R/W	0000_0000h	56.3.15/ 2780
E8	PSI5 Message Register High i (PSI5_CH0_PMRH21)	32	R/W	0000_0000h	56.3.16/ 2781
EC	PSI5 Message Register Low i (PSI5_CH0_PMRL22)	32	R/W	0000_0000h	56.3.15/ 2780
F0	PSI5 Message Register High i (PSI5_CH0_PMRH22)	32	R/W	0000_0000h	56.3.16/ 2781
F4	PSI5 Message Register Low i (PSI5_CH0_PMRL23)	32	R/W	0000_0000h	56.3.15/ 2780
F8	PSI5 Message Register High i (PSI5_CH0_PMRH23)	32	R/W	0000_0000h	56.3.16/ 2781
FC	PSI5 Message Register Low i (PSI5_CH0_PMRL24)	32	R/W	0000_0000h	56.3.15/ 2780
100	PSI5 Message Register High i (PSI5_CH0_PMRH24)	32	R/W	0000_0000h	56.3.16/ 2781
104	PSI5 Message Register Low i (PSI5_CH0_PMRL25)	32	R/W	0000_0000h	56.3.15/ 2780
108	PSI5 Message Register High i (PSI5_CH0_PMRH25)	32	R/W	0000_0000h	56.3.16/ 2781
10C	PSI5 Message Register Low i (PSI5_CH0_PMRL26)	32	R/W	0000_0000h	56.3.15/ 2780
110	PSI5 Message Register High i (PSI5_CH0_PMRH26)	32	R/W	0000_0000h	56.3.16/ 2781
114	PSI5 Message Register Low i (PSI5_CH0_PMRL27)	32	R/W	0000_0000h	56.3.15/ 2780
118	PSI5 Message Register High i (PSI5_CH0_PMRH27)	32	R/W	0000_0000h	56.3.16/ 2781
11C	PSI5 Message Register Low i (PSI5_CH0_PMRL28)	32	R/W	0000_0000h	56.3.15/ 2780
120	PSI5 Message Register High i (PSI5_CH0_PMRH28)	32	R/W	0000_0000h	56.3.16/ 2781
124	PSI5 Message Register Low i (PSI5_CH0_PMRL29)	32	R/W	0000_0000h	56.3.15/ 2780
128	PSI5 Message Register High i (PSI5_CH0_PMRH29)	32	R/W	0000_0000h	56.3.16/ 2781
12C	PSI5 Message Register Low i (PSI5_CH0_PMRL30)	32	R/W	0000_0000h	56.3.15/ 2780

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
130	PSI5 Message Register High i (PSI5_CH0_PMRH30)	32	R/W	0000_0000h	56.3.16/ 2781
134	PSI5 Message Register Low i (PSI5_CH0_PMRL31)	32	R/W	0000_0000h	56.3.15/ 2780
138	PSI5 Message Register High i (PSI5_CH0_PMRH31)	32	R/W	0000_0000h	56.3.16/ 2781
13C	SMC Frame Register n (PSI5_CH0_SFR1)	32	R/W	0000_0000h	56.3.17/ 2782
140	SMC Frame Register n (PSI5_CH0_SFR2)	32	R/W	0000_0000h	56.3.17/ 2782
144	SMC Frame Register n (PSI5_CH0_SFR3)	32	R/W	0000_0000h	56.3.17/ 2782
148	SMC Frame Register n (PSI5_CH0_SFR4)	32	R/W	0000_0000h	56.3.17/ 2782
14C	SMC Frame Register n (PSI5_CH0_SFR5)	32	R/W	0000_0000h	56.3.17/ 2782
150	SMC Frame Register n (PSI5_CH0_SFR6)	32	R/W	0000_0000h	56.3.17/ 2782
154	New Data Status Register (PSI5_CH0_NDSR)	32	w1c	0000_0000h	56.3.18/ 2784
158	Overwrite Status Register (PSI5_CH0_OWSR)	32	w1c	0000_0000h	56.3.19/ 2785
15C	Error Indication Status Register (PSI5_CH0_EISR)	32	w1c	0000_0000h	56.3.20/ 2785
160	Set New Data Status Register (PSI5_CH0_SNDSR)	32	W	0000_0000h	56.3.21/ 2786
164	Set Overwrite Status Register (PSI5_CH0_SOWSR)	32	W	0000_0000h	56.3.22/ 2786
168	Set Error Status Register (PSI5_CH0_SEISR)	32	W	0000_0000h	56.3.23/ 2787
16C	Set SMC Error Status Register (PSI5_CH0_SSESR)	32	W	0000_0000h	56.3.24/ 2787
170	Sync Time Stamp Read Register (PSI5_CH0_STSRR)	32	R	0000_0000h	56.3.25/ 2788
174	Data Time Stamp Read Register (PSI5_CH0_DTSRR)	32	R	0000_0000h	56.3.26/ 2789
178	Slot n Frame Configuration Register (PSI5_CH0_S1FCR)	32	R/W	0000_0010h	56.3.27/ 2789
17C	Slot n Frame Configuration Register (PSI5_CH0_S2FCR)	32	R/W	0000_0010h	56.3.27/ 2789
180	Slot n Frame Configuration Register (PSI5_CH0_S3FCR)	32	R/W	0000_0010h	56.3.27/ 2789
184	Slot n Frame Configuration Register (PSI5_CH0_S4FCR)	32	R/W	0000_0010h	56.3.27/ 2789

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
188	Slot n Frame Configuration Register (PSI5_CH0_S5FCR)	32	R/W	0000_0010h	56.3.27/ 2789
18C	Slot n Frame Configuration Register (PSI5_CH0_S6FCR)	32	R/W	0000_0010h	56.3.27/ 2789
190	Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)	16	R	0000h	56.3.28/ 2791
192	Slot 2 Start Boundary Register (PSI5_CH0_S2SBR)	16	R	0000h	56.3.29/ 2791
194	Slot 3 Start Boundary Register (PSI5_CH0_S3SBR)	16	R	0000h	56.3.30/ 2792
196	Slot 4 Start Boundary Register (PSI5_CH0_S4SBR)	16	R	0000h	56.3.31/ 2792
198	Slot 5 Start Boundary Register (PSI5_CH0_S5SBR)	16	R	0000h	56.3.32/ 2793
19A	Slot 6 Start Boundary Register (PSI5_CH0_S6SBR)	16	R	0000h	56.3.33/ 2793
19C	Slot n End Boundary Register (PSI5_CH0_SnEBR)	32	R/W	0000_0000h	56.3.34/ 2794
1A0	Data Output Block Configuration Register (PSI5_CH0_DOBCR)	16	R/W	0000h	56.3.35/ 2795
1A2	Manchestor Decoder Disable Offset (PSI5_CH0_MDDIS_OFF)	16	R/W	0000h	56.3.36/ 2798
1A4	Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D)	16	R/W	0000h	56.3.37/ 2799
1A6	Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D)	16	R/W	0000h	56.3.38/ 2799
1A8	Counter Target Pulse Register (PSI5_CH0_CTPR)	16	R/W	0000h	56.3.39/ 2800
1AA	Counter Initialize Pulse Register (PSI5_CH0_CIPR)	16	R/W	0000h	56.3.40/ 2801
1AC	Data Preparation Register Low (PSI5_CH0_DPRL)	32	R/W	0000_0000h	56.3.41/ 2801
1B0	Data Preparation Register High (PSI5_CH0_DPRH)	32	R	0000_0000h	56.3.42/ 2802
1B4	Data Buffer Register Low (PSI5_CH0_DBRL)	32	R/W	0000_0000h	56.3.43/ 2803
1B8	Data Buffer Register High (PSI5_CH0_DBRH)	32	R/W	0000_0000h	56.3.44/ 2805
1BC	Data Shift Register Low (PSI5_CH0_DSRL)	32	R/W	0000_0000h	56.3.45/ 2806
1C0	Data Shift Register High (PSI5_CH0_DSRH)	32	R/W	0000_0000h	56.3.46/ 2807
1C8	PSI5 Channel Control Register (PSI5_CH1_PCCR)	32	R/W	001F_0000h	56.3.47/ 2809

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1CC	DMA Control Register (PSI5_CH1_DCR)	32	R/W	0000_0000h	56.3.48/ 2814
1D0	DMA Status Register (PSI5_CH1_DSR)	32	w1c	0000_0000h	56.3.49/ 2817
1D4	General Interrupt Control Register (PSI5_CH1_GICR)	32	R/W	0000_0000h	56.3.50/ 2819
1D8	New Data Interrupt Control Register (PSI5_CH1_NDICR)	32	R/W	0000_0000h	56.3.51/ 2821
1DC	Overwrite Interrupt Control Register (PSI5_CH1_OWICR)	32	R/W	0000_0000h	56.3.52/ 2822
1E0	Error Interrupt Control Register (PSI5_CH1_EICR)	32	R/W	0000_0000h	56.3.53/ 2822
1E4	General Interrupt Status Register (PSI5_CH1_GISR)	32	w1c	0007_0000h	56.3.54/ 2823
1E8	DMA PSI5 Message Register (PSI5_CH1_DPMR)	32	R	0000_0000h	56.3.55/ 2825
1EC	DMA SMC Frame Register (PSI5_CH1_DSFR)	32	R	0000_0000h	56.3.56/ 2826
1F0	DMA Diagnostic Status Register (PSI5_CH1_DDSR)	32	R	0000_0000h	56.3.57/ 2826
1F4	PSI5 Message Receive Register Low (PSI5_CH1_PMRRL)	32	R	0000_0000h	56.3.58/ 2827
1F8	PSI5 Message Receive Register High (PSI5_CH1_PMRRH)	32	R	0000_0000h	56.3.59/ 2828
1FC	PSI5 Message Register Low i (PSI5_CH1_PMRL0)	32	R/W	0000_0000h	56.3.60/ 2829
200	PSI5 Message Register High i (PSI5_CH1_PMRH0)	32	R/W	0000_0000h	56.3.61/ 2830
204	PSI5 Message Register Low i (PSI5_CH1_PMRL1)	32	R/W	0000_0000h	56.3.60/ 2829
208	PSI5 Message Register High i (PSI5_CH1_PMRH1)	32	R/W	0000_0000h	56.3.61/ 2830
20C	PSI5 Message Register Low i (PSI5_CH1_PMRL2)	32	R/W	0000_0000h	56.3.60/ 2829
210	PSI5 Message Register High i (PSI5_CH1_PMRH2)	32	R/W	0000_0000h	56.3.61/ 2830
214	PSI5 Message Register Low i (PSI5_CH1_PMRL3)	32	R/W	0000_0000h	56.3.60/ 2829
218	PSI5 Message Register High i (PSI5_CH1_PMRH3)	32	R/W	0000_0000h	56.3.61/ 2830
21C	PSI5 Message Register Low i (PSI5_CH1_PMRL4)	32	R/W	0000_0000h	56.3.60/ 2829
220	PSI5 Message Register High i (PSI5_CH1_PMRH4)	32	R/W	0000_0000h	56.3.61/ 2830

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
224	PSI5 Message Register Low i (PSI5_CH1_PMRL5)	32	R/W	0000_0000h	56.3.60/ 2829
228	PSI5 Message Register High i (PSI5_CH1_PMRH5)	32	R/W	0000_0000h	56.3.61/ 2830
22C	PSI5 Message Register Low i (PSI5_CH1_PMRL6)	32	R/W	0000_0000h	56.3.60/ 2829
230	PSI5 Message Register High i (PSI5_CH1_PMRH6)	32	R/W	0000_0000h	56.3.61/ 2830
234	PSI5 Message Register Low i (PSI5_CH1_PMRL7)	32	R/W	0000_0000h	56.3.60/ 2829
238	PSI5 Message Register High i (PSI5_CH1_PMRH7)	32	R/W	0000_0000h	56.3.61/ 2830
23C	PSI5 Message Register Low i (PSI5_CH1_PMRL8)	32	R/W	0000_0000h	56.3.60/ 2829
240	PSI5 Message Register High i (PSI5_CH1_PMRH8)	32	R/W	0000_0000h	56.3.61/ 2830
244	PSI5 Message Register Low i (PSI5_CH1_PMRL9)	32	R/W	0000_0000h	56.3.60/ 2829
248	PSI5 Message Register High i (PSI5_CH1_PMRH9)	32	R/W	0000_0000h	56.3.61/ 2830
24C	PSI5 Message Register Low i (PSI5_CH1_PMRL10)	32	R/W	0000_0000h	56.3.60/ 2829
250	PSI5 Message Register High i (PSI5_CH1_PMRH10)	32	R/W	0000_0000h	56.3.61/ 2830
254	PSI5 Message Register Low i (PSI5_CH1_PMRL11)	32	R/W	0000_0000h	56.3.60/ 2829
258	PSI5 Message Register High i (PSI5_CH1_PMRH11)	32	R/W	0000_0000h	56.3.61/ 2830
25C	PSI5 Message Register Low i (PSI5_CH1_PMRL12)	32	R/W	0000_0000h	56.3.60/ 2829
260	PSI5 Message Register High i (PSI5_CH1_PMRH12)	32	R/W	0000_0000h	56.3.61/ 2830
264	PSI5 Message Register Low i (PSI5_CH1_PMRL13)	32	R/W	0000_0000h	56.3.60/ 2829
268	PSI5 Message Register High i (PSI5_CH1_PMRH13)	32	R/W	0000_0000h	56.3.61/ 2830
26C	PSI5 Message Register Low i (PSI5_CH1_PMRL14)	32	R/W	0000_0000h	56.3.60/ 2829
270	PSI5 Message Register High i (PSI5_CH1_PMRH14)	32	R/W	0000_0000h	56.3.61/ 2830
274	PSI5 Message Register Low i (PSI5_CH1_PMRL15)	32	R/W	0000_0000h	56.3.60/ 2829
278	PSI5 Message Register High i (PSI5_CH1_PMRH15)	32	R/W	0000_0000h	56.3.61/ 2830

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
27C	PSI5 Message Register Low i (PSI5_CH1_PMRL16)	32	R/W	0000_0000h	56.3.60/ 2829
280	PSI5 Message Register High i (PSI5_CH1_PMRH16)	32	R/W	0000_0000h	56.3.61/ 2830
284	PSI5 Message Register Low i (PSI5_CH1_PMRL17)	32	R/W	0000_0000h	56.3.60/ 2829
288	PSI5 Message Register High i (PSI5_CH1_PMRH17)	32	R/W	0000_0000h	56.3.61/ 2830
28C	PSI5 Message Register Low i (PSI5_CH1_PMRL18)	32	R/W	0000_0000h	56.3.60/ 2829
290	PSI5 Message Register High i (PSI5_CH1_PMRH18)	32	R/W	0000_0000h	56.3.61/ 2830
294	PSI5 Message Register Low i (PSI5_CH1_PMRL19)	32	R/W	0000_0000h	56.3.60/ 2829
298	PSI5 Message Register High i (PSI5_CH1_PMRH19)	32	R/W	0000_0000h	56.3.61/ 2830
29C	PSI5 Message Register Low i (PSI5_CH1_PMRL20)	32	R/W	0000_0000h	56.3.60/ 2829
2A0	PSI5 Message Register High i (PSI5_CH1_PMRH20)	32	R/W	0000_0000h	56.3.61/ 2830
2A4	PSI5 Message Register Low i (PSI5_CH1_PMRL21)	32	R/W	0000_0000h	56.3.60/ 2829
2A8	PSI5 Message Register High i (PSI5_CH1_PMRH21)	32	R/W	0000_0000h	56.3.61/ 2830
2AC	PSI5 Message Register Low i (PSI5_CH1_PMRL22)	32	R/W	0000_0000h	56.3.60/ 2829
2B0	PSI5 Message Register High i (PSI5_CH1_PMRH22)	32	R/W	0000_0000h	56.3.61/ 2830
2B4	PSI5 Message Register Low i (PSI5_CH1_PMRL23)	32	R/W	0000_0000h	56.3.60/ 2829
2B8	PSI5 Message Register High i (PSI5_CH1_PMRH23)	32	R/W	0000_0000h	56.3.61/ 2830
2BC	PSI5 Message Register Low i (PSI5_CH1_PMRL24)	32	R/W	0000_0000h	56.3.60/ 2829
2C0	PSI5 Message Register High i (PSI5_CH1_PMRH24)	32	R/W	0000_0000h	56.3.61/ 2830
2C4	PSI5 Message Register Low i (PSI5_CH1_PMRL25)	32	R/W	0000_0000h	56.3.60/ 2829
2C8	PSI5 Message Register High i (PSI5_CH1_PMRH25)	32	R/W	0000_0000h	56.3.61/ 2830
2CC	PSI5 Message Register Low i (PSI5_CH1_PMRL26)	32	R/W	0000_0000h	56.3.60/ 2829
2D0	PSI5 Message Register High i (PSI5_CH1_PMRH26)	32	R/W	0000_0000h	56.3.61/ 2830

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2D4	PSI5 Message Register Low i (PSI5_CH1_PMRL27)	32	R/W	0000_0000h	56.3.60/ 2829
2D8	PSI5 Message Register High i (PSI5_CH1_PMRH27)	32	R/W	0000_0000h	56.3.61/ 2830
2DC	PSI5 Message Register Low i (PSI5_CH1_PMRL28)	32	R/W	0000_0000h	56.3.60/ 2829
2E0	PSI5 Message Register High i (PSI5_CH1_PMRH28)	32	R/W	0000_0000h	56.3.61/ 2830
2E4	PSI5 Message Register Low i (PSI5_CH1_PMRL29)	32	R/W	0000_0000h	56.3.60/ 2829
2E8	PSI5 Message Register High i (PSI5_CH1_PMRH29)	32	R/W	0000_0000h	56.3.61/ 2830
2EC	PSI5 Message Register Low i (PSI5_CH1_PMRL30)	32	R/W	0000_0000h	56.3.60/ 2829
2F0	PSI5 Message Register High i (PSI5_CH1_PMRH30)	32	R/W	0000_0000h	56.3.61/ 2830
2F4	PSI5 Message Register Low i (PSI5_CH1_PMRL31)	32	R/W	0000_0000h	56.3.60/ 2829
2F8	PSI5 Message Register High i (PSI5_CH1_PMRH31)	32	R/W	0000_0000h	56.3.61/ 2830
2FC	SMC Frame Register n (PSI5_CH1_SFR1)	32	R/W	0000_0000h	56.3.62/ 2831
300	SMC Frame Register n (PSI5_CH1_SFR2)	32	R/W	0000_0000h	56.3.62/ 2831
304	SMC Frame Register n (PSI5_CH1_SFR3)	32	R/W	0000_0000h	56.3.62/ 2831
308	SMC Frame Register n (PSI5_CH1_SFR4)	32	R/W	0000_0000h	56.3.62/ 2831
30C	SMC Frame Register n (PSI5_CH1_SFR5)	32	R/W	0000_0000h	56.3.62/ 2831
310	SMC Frame Register n (PSI5_CH1_SFR6)	32	R/W	0000_0000h	56.3.62/ 2831
314	New Data Status Register (PSI5_CH1_NDSR)	32	w1c	0000_0000h	56.3.63/ 2833
318	Overwrite Status Register (PSI5_CH1_OWSR)	32	w1c	0000_0000h	56.3.64/ 2834
31C	Error Indication Status Register (PSI5_CH1_EISR)	32	w1c	0000_0000h	56.3.65/ 2834
320	Set New Data Status Register (PSI5_CH1_SNDSR)	32	W	0000_0000h	56.3.66/ 2835
324	Set Overwrite Status Register (PSI5_CH1_SOWSR)	32	W	0000_0000h	56.3.67/ 2835
328	Set Error Status Register (PSI5_CH1_SEISR)	32	W	0000_0000h	56.3.68/ 2836

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
32C	Set SMC Error Status Register (PSI5_CH1_SSESR)	32	W	0000_0000h	56.3.69/ 2836
330	Sync Time Stamp Read Register (PSI5_CH1_STSRR)	32	R	0000_0000h	56.3.70/ 2837
334	Data Time Stamp Read Register (PSI5_CH1_DTSRR)	32	R	0000_0000h	56.3.71/ 2838
338	Slot n Frame Configuration Register (PSI5_CH1_S1FCR)	32	R/W	0000_0010h	56.3.72/ 2838
33C	Slot n Frame Configuration Register (PSI5_CH1_S2FCR)	32	R/W	0000_0010h	56.3.72/ 2838
340	Slot n Frame Configuration Register (PSI5_CH1_S3FCR)	32	R/W	0000_0010h	56.3.72/ 2838
344	Slot n Frame Configuration Register (PSI5_CH1_S4FCR)	32	R/W	0000_0010h	56.3.72/ 2838
348	Slot n Frame Configuration Register (PSI5_CH1_S5FCR)	32	R/W	0000_0010h	56.3.72/ 2838
34C	Slot n Frame Configuration Register (PSI5_CH1_S6FCR)	32	R/W	0000_0010h	56.3.72/ 2838
350	Slot 1 Start Boundary Register (PSI5_CH1_S1SBR)	16	R	0000h	56.3.73/ 2840
352	Slot 2 Start Boundary Register (PSI5_CH1_S2SBR)	16	R	0000h	56.3.74/ 2840
354	Slot 3 Start Boundary Register (PSI5_CH1_S3SBR)	16	R	0000h	56.3.75/ 2841
356	Slot 4 Start Boundary Register (PSI5_CH1_S4SBR)	16	R	0000h	56.3.76/ 2841
358	Slot 5 Start Boundary Register (PSI5_CH1_S5SBR)	16	R	0000h	56.3.77/ 2842
35A	Slot 6 Start Boundary Register (PSI5_CH1_S6SBR)	16	R	0000h	56.3.78/ 2842
35C	Slot n End Boundary Register (PSI5_CH1_SnEBR)	32	R/W	0000_0000h	56.3.79/ 2843
360	Data Output Block Configuration Register (PSI5_CH1_DOBCR)	16	R/W	0000h	56.3.80/ 2844
362	Manchestor Decoder Disable Offset (PSI5_CH1_MDDIS_OFF)	16	R/W	0000h	56.3.81/ 2847
364	Pulse Width for Data Bit Value 0 (PSI5_CH1_PW0D)	16	R/W	0000h	56.3.82/ 2848
366	Pulse Width for Data Bit Value 1 (PSI5_CH1_PW1D)	16	R/W	0000h	56.3.83/ 2848
368	Counter Target Pulse Register (PSI5_CH1_CTPR)	16	R/W	0000h	56.3.84/ 2849
36A	Counter Initialize Pulse Register (PSI5_CH1_CIPR)	16	R/W	0000h	56.3.85/ 2850

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
36C	Data Preparation Register Low (PSI5_CH1_DPRL)	32	R/W	0000_0000h	56.3.86/ 2850
370	Data Preparation Register High (PSI5_CH1_DPRH)	32	R	0000_0000h	56.3.87/ 2851
374	Data Buffer Register Low (PSI5_CH1_DBRL)	32	R/W	0000_0000h	56.3.88/ 2852
378	Data Buffer Register High (PSI5_CH1_DBRH)	32	R/W	0000_0000h	56.3.89/ 2854
37C	Data Shift Register Low (PSI5_CH1_DSRL)	32	R/W	0000_0000h	56.3.90/ 2855
380	Data Shift Register High (PSI5_CH1_DSRH)	32	R/W	0000_0000h	56.3.91/ 2856
388	PSI5 Channel Control Register (PSI5_CH2_PCCR)	32	R/W	001F_0000h	56.3.92/ 2858
38C	DMA Control Register (PSI5_CH2_DCR)	32	R/W	0000_0000h	56.3.93/ 2863
390	DMA Status Register (PSI5_CH2_DSR)	32	w1c	0000_0000h	56.3.94/ 2866
394	General Interrupt Control Register (PSI5_CH2_GICR)	32	R/W	0000_0000h	56.3.95/ 2868
398	New Data Interrupt Control Register (PSI5_CH2_NDICR)	32	R/W	0000_0000h	56.3.96/ 2870
39C	Overwrite Interrupt Control Register (PSI5_CH2_OWICR)	32	R/W	0000_0000h	56.3.97/ 2871
3A0	Error Interrupt Control Register (PSI5_CH2_EICR)	32	R/W	0000_0000h	56.3.98/ 2871
3A4	General Interrupt Status Register (PSI5_CH2_GISR)	32	w1c	0007_0000h	56.3.99/ 2872
3A8	DMA PSI5 Message Register (PSI5_CH2_DPMR)	32	R	0000_0000h	56.3.100/ 2874
3AC	DMA SMC Frame Register (PSI5_CH2_DSFR)	32	R	0000_0000h	56.3.101/ 2875
3B0	DMA Diagnostic Status Register (PSI5_CH2_DDSR)	32	R	0000_0000h	56.3.102/ 2875
3B4	PSI5 Message Receive Register Low (PSI5_CH2_PMRRL)	32	R	0000_0000h	56.3.103/ 2876
3B8	PSI5 Message Receive Register High (PSI5_CH2_PMRRH)	32	R	0000_0000h	56.3.104/ 2877
3BC	PSI5 Message Register Low i (PSI5_CH2_PMRL0)	32	R/W	0000_0000h	56.3.105/ 2878
3C0	PSI5 Message Register High i (PSI5_CH2_PMRH0)	32	R/W	0000_0000h	56.3.106/ 2879
3C4	PSI5 Message Register Low i (PSI5_CH2_PMRL1)	32	R/W	0000_0000h	56.3.105/ 2878

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3C8	PSI5 Message Register High i (PSI5_CH2_PMRH1)	32	R/W	0000_0000h	56.3.106/ 2879
3CC	PSI5 Message Register Low i (PSI5_CH2_PMRL2)	32	R/W	0000_0000h	56.3.105/ 2878
3D0	PSI5 Message Register High i (PSI5_CH2_PMRH2)	32	R/W	0000_0000h	56.3.106/ 2879
3D4	PSI5 Message Register Low i (PSI5_CH2_PMRL3)	32	R/W	0000_0000h	56.3.105/ 2878
3D8	PSI5 Message Register High i (PSI5_CH2_PMRH3)	32	R/W	0000_0000h	56.3.106/ 2879
3DC	PSI5 Message Register Low i (PSI5_CH2_PMRL4)	32	R/W	0000_0000h	56.3.105/ 2878
3E0	PSI5 Message Register High i (PSI5_CH2_PMRH4)	32	R/W	0000_0000h	56.3.106/ 2879
3E4	PSI5 Message Register Low i (PSI5_CH2_PMRL5)	32	R/W	0000_0000h	56.3.105/ 2878
3E8	PSI5 Message Register High i (PSI5_CH2_PMRH5)	32	R/W	0000_0000h	56.3.106/ 2879
3EC	PSI5 Message Register Low i (PSI5_CH2_PMRL6)	32	R/W	0000_0000h	56.3.105/ 2878
3F0	PSI5 Message Register High i (PSI5_CH2_PMRH6)	32	R/W	0000_0000h	56.3.106/ 2879
3F4	PSI5 Message Register Low i (PSI5_CH2_PMRL7)	32	R/W	0000_0000h	56.3.105/ 2878
3F8	PSI5 Message Register High i (PSI5_CH2_PMRH7)	32	R/W	0000_0000h	56.3.106/ 2879
3FC	PSI5 Message Register Low i (PSI5_CH2_PMRL8)	32	R/W	0000_0000h	56.3.105/ 2878
400	PSI5 Message Register High i (PSI5_CH2_PMRH8)	32	R/W	0000_0000h	56.3.106/ 2879
404	PSI5 Message Register Low i (PSI5_CH2_PMRL9)	32	R/W	0000_0000h	56.3.105/ 2878
408	PSI5 Message Register High i (PSI5_CH2_PMRH9)	32	R/W	0000_0000h	56.3.106/ 2879
40C	PSI5 Message Register Low i (PSI5_CH2_PMRL10)	32	R/W	0000_0000h	56.3.105/ 2878
410	PSI5 Message Register High i (PSI5_CH2_PMRH10)	32	R/W	0000_0000h	56.3.106/ 2879
414	PSI5 Message Register Low i (PSI5_CH2_PMRL11)	32	R/W	0000_0000h	56.3.105/ 2878
418	PSI5 Message Register High i (PSI5_CH2_PMRH11)	32	R/W	0000_0000h	56.3.106/ 2879
41C	PSI5 Message Register Low i (PSI5_CH2_PMRL12)	32	R/W	0000_0000h	56.3.105/ 2878

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
420	PSI5 Message Register High i (PSI5_CH2_PMRH12)	32	R/W	0000_0000h	56.3.106/ 2879
424	PSI5 Message Register Low i (PSI5_CH2_PMRL13)	32	R/W	0000_0000h	56.3.105/ 2878
428	PSI5 Message Register High i (PSI5_CH2_PMRH13)	32	R/W	0000_0000h	56.3.106/ 2879
42C	PSI5 Message Register Low i (PSI5_CH2_PMRL14)	32	R/W	0000_0000h	56.3.105/ 2878
430	PSI5 Message Register High i (PSI5_CH2_PMRH14)	32	R/W	0000_0000h	56.3.106/ 2879
434	PSI5 Message Register Low i (PSI5_CH2_PMRL15)	32	R/W	0000_0000h	56.3.105/ 2878
438	PSI5 Message Register High i (PSI5_CH2_PMRH15)	32	R/W	0000_0000h	56.3.106/ 2879
43C	PSI5 Message Register Low i (PSI5_CH2_PMRL16)	32	R/W	0000_0000h	56.3.105/ 2878
440	PSI5 Message Register High i (PSI5_CH2_PMRH16)	32	R/W	0000_0000h	56.3.106/ 2879
444	PSI5 Message Register Low i (PSI5_CH2_PMRL17)	32	R/W	0000_0000h	56.3.105/ 2878
448	PSI5 Message Register High i (PSI5_CH2_PMRH17)	32	R/W	0000_0000h	56.3.106/ 2879
44C	PSI5 Message Register Low i (PSI5_CH2_PMRL18)	32	R/W	0000_0000h	56.3.105/ 2878
450	PSI5 Message Register High i (PSI5_CH2_PMRH18)	32	R/W	0000_0000h	56.3.106/ 2879
454	PSI5 Message Register Low i (PSI5_CH2_PMRL19)	32	R/W	0000_0000h	56.3.105/ 2878
458	PSI5 Message Register High i (PSI5_CH2_PMRH19)	32	R/W	0000_0000h	56.3.106/ 2879
45C	PSI5 Message Register Low i (PSI5_CH2_PMRL20)	32	R/W	0000_0000h	56.3.105/ 2878
460	PSI5 Message Register High i (PSI5_CH2_PMRH20)	32	R/W	0000_0000h	56.3.106/ 2879
464	PSI5 Message Register Low i (PSI5_CH2_PMRL21)	32	R/W	0000_0000h	56.3.105/ 2878
468	PSI5 Message Register High i (PSI5_CH2_PMRH21)	32	R/W	0000_0000h	56.3.106/ 2879
46C	PSI5 Message Register Low i (PSI5_CH2_PMRL22)	32	R/W	0000_0000h	56.3.105/ 2878
470	PSI5 Message Register High i (PSI5_CH2_PMRH22)	32	R/W	0000_0000h	56.3.106/ 2879
474	PSI5 Message Register Low i (PSI5_CH2_PMRL23)	32	R/W	0000_0000h	56.3.105/ 2878

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
478	PSI5 Message Register High i (PSI5_CH2_PMRH23)	32	R/W	0000_0000h	56.3.106/ 2879
47C	PSI5 Message Register Low i (PSI5_CH2_PMRL24)	32	R/W	0000_0000h	56.3.105/ 2878
480	PSI5 Message Register High i (PSI5_CH2_PMRH24)	32	R/W	0000_0000h	56.3.106/ 2879
484	PSI5 Message Register Low i (PSI5_CH2_PMRL25)	32	R/W	0000_0000h	56.3.105/ 2878
488	PSI5 Message Register High i (PSI5_CH2_PMRH25)	32	R/W	0000_0000h	56.3.106/ 2879
48C	PSI5 Message Register Low i (PSI5_CH2_PMRL26)	32	R/W	0000_0000h	56.3.105/ 2878
490	PSI5 Message Register High i (PSI5_CH2_PMRH26)	32	R/W	0000_0000h	56.3.106/ 2879
494	PSI5 Message Register Low i (PSI5_CH2_PMRL27)	32	R/W	0000_0000h	56.3.105/ 2878
498	PSI5 Message Register High i (PSI5_CH2_PMRH27)	32	R/W	0000_0000h	56.3.106/ 2879
49C	PSI5 Message Register Low i (PSI5_CH2_PMRL28)	32	R/W	0000_0000h	56.3.105/ 2878
4A0	PSI5 Message Register High i (PSI5_CH2_PMRH28)	32	R/W	0000_0000h	56.3.106/ 2879
4A4	PSI5 Message Register Low i (PSI5_CH2_PMRL29)	32	R/W	0000_0000h	56.3.105/ 2878
4A8	PSI5 Message Register High i (PSI5_CH2_PMRH29)	32	R/W	0000_0000h	56.3.106/ 2879
4AC	PSI5 Message Register Low i (PSI5_CH2_PMRL30)	32	R/W	0000_0000h	56.3.105/ 2878
4B0	PSI5 Message Register High i (PSI5_CH2_PMRH30)	32	R/W	0000_0000h	56.3.106/ 2879
4B4	PSI5 Message Register Low i (PSI5_CH2_PMRL31)	32	R/W	0000_0000h	56.3.105/ 2878
4B8	PSI5 Message Register High i (PSI5_CH2_PMRH31)	32	R/W	0000_0000h	56.3.106/ 2879
4BC	SMC Frame Register n (PSI5_CH2_SFR1)	32	R/W	0000_0000h	56.3.107/ 2880
4C0	SMC Frame Register n (PSI5_CH2_SFR2)	32	R/W	0000_0000h	56.3.107/ 2880
4C4	SMC Frame Register n (PSI5_CH2_SFR3)	32	R/W	0000_0000h	56.3.107/ 2880
4C8	SMC Frame Register n (PSI5_CH2_SFR4)	32	R/W	0000_0000h	56.3.107/ 2880
4CC	SMC Frame Register n (PSI5_CH2_SFR5)	32	R/W	0000_0000h	56.3.107/ 2880

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4D0	SMC Frame Register n (PSI5_CH2_SFR6)	32	R/W	0000_0000h	56.3.107/ 2880
4D4	New Data Status Register (PSI5_CH2_NDSR)	32	w1c	0000_0000h	56.3.108/ 2882
4D8	Overwrite Status Register (PSI5_CH2_OWSR)	32	w1c	0000_0000h	56.3.109/ 2883
4DC	Error Indication Status Register (PSI5_CH2_EISR)	32	w1c	0000_0000h	56.3.110/ 2883
4E0	Set New Data Status Register (PSI5_CH2_SNDSR)	32	W	0000_0000h	56.3.111/ 2884
4E4	Set Overwrite Status Register (PSI5_CH2_SOWSR)	32	W	0000_0000h	56.3.112/ 2884
4E8	Set Error Status Register (PSI5_CH2_SEISR)	32	W	0000_0000h	56.3.113/ 2885
4EC	Set SMC Error Status Register (PSI5_CH2_SSESR)	32	W	0000_0000h	56.3.114/ 2885
4F0	Sync Time Stamp Read Register (PSI5_CH2_STSRR)	32	R	0000_0000h	56.3.115/ 2886
4F4	Data Time Stamp Read Register (PSI5_CH2_DTSRR)	32	R	0000_0000h	56.3.116/ 2887
4F8	Slot n Frame Configuration Register (PSI5_CH2_S1FCR)	32	R/W	0000_0010h	56.3.117/ 2887
4FC	Slot n Frame Configuration Register (PSI5_CH2_S2FCR)	32	R/W	0000_0010h	56.3.117/ 2887
500	Slot n Frame Configuration Register (PSI5_CH2_S3FCR)	32	R/W	0000_0010h	56.3.117/ 2887
504	Slot n Frame Configuration Register (PSI5_CH2_S4FCR)	32	R/W	0000_0010h	56.3.117/ 2887
508	Slot n Frame Configuration Register (PSI5_CH2_S5FCR)	32	R/W	0000_0010h	56.3.117/ 2887
50C	Slot n Frame Configuration Register (PSI5_CH2_S6FCR)	32	R/W	0000_0010h	56.3.117/ 2887
510	Slot 1 Start Boundary Register (PSI5_CH2_S1SBR)	16	R	0000h	56.3.118/ 2889
512	Slot 2 Start Boundary Register (PSI5_CH2_S2SBR)	16	R	0000h	56.3.119/ 2889
514	Slot 3 Start Boundary Register (PSI5_CH2_S3SBR)	16	R	0000h	56.3.120/ 2890
516	Slot 4 Start Boundary Register (PSI5_CH2_S4SBR)	16	R	0000h	56.3.121/ 2890
518	Slot 5 Start Boundary Register (PSI5_CH2_S5SBR)	16	R	0000h	56.3.122/ 2891
51A	Slot 6 Start Boundary Register (PSI5_CH2_S6SBR)	16	R	0000h	56.3.123/ 2891

Table continues on the next page...

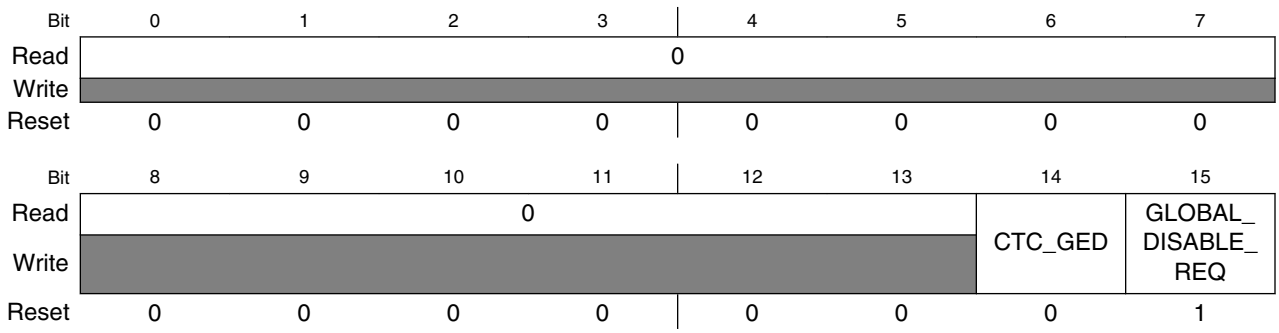
PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
51C	Slot n End Boundary Register (PSI5_CH2_SnEBR)	32	R/W	0000_0000h	56.3.124/ 2892
520	Data Output Block Configuration Register (PSI5_CH2_DOBCR)	16	R/W	0000h	56.3.125/ 2893
522	Manchestor Decoder Disable Offset (PSI5_CH2_MDDIS_OFF)	16	R/W	0000h	56.3.126/ 2896
524	Pulse Width for Data Bit Value 0 (PSI5_CH2_PW0D)	16	R/W	0000h	56.3.127/ 2897
526	Pulse Width for Data Bit Value 1 (PSI5_CH2_PW1D)	16	R/W	0000h	56.3.128/ 2897
528	Counter Target Pulse Register (PSI5_CH2_CTPR)	16	R/W	0000h	56.3.129/ 2898
52A	Counter Initialize Pulse Register (PSI5_CH2_CIPR)	16	R/W	0000h	56.3.130/ 2899
52C	Data Preparation Register Low (PSI5_CH2_DPRL)	32	R/W	0000_0000h	56.3.131/ 2899
530	Data Preparation Register High (PSI5_CH2_DPRH)	32	R	0000_0000h	56.3.132/ 2900
534	Data Buffer Register Low (PSI5_CH2_DBRL)	32	R/W	0000_0000h	56.3.133/ 2901
538	Data Buffer Register High (PSI5_CH2_DBRH)	32	R/W	0000_0000h	56.3.134/ 2903
53C	Data Shift Register Low (PSI5_CH2_DSRL)	32	R/W	0000_0000h	56.3.135/ 2904
540	Data Shift Register High (PSI5_CH2_DSRH)	32	R/W	0000_0000h	56.3.136/ 2905

56.3.1 Global Control Register (PSI5_GCR)

The Global Control Register (PSI5_GCR)[1] provides control functions for all the PSI5 channels simultaneously.

Address: 0h base + 0h offset = 0h



PSI5_GCR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CTC_GED	Channel Target Counter (CTC) Global Enable/Disable. **NOTE:** This bit is writable in all modes. 0 All the CTC counters having PCCR[CTC_GED_SEL] = 1 are reset. 1 Simultaneously starts the CTC of all the channels that have PCCR[CTC_GED_SEL] = 1.
15 GLOBAL_ DISABLE_REQ	PSI5 Receive Global disable request. When set, all PSI5 channels enter Disable mode (from individual Normal or Config modes) regardless of the settings of the PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN] bits. When cleared, each PSI5 channel enters Config mode or Normal mode based on the setting of the PCCR[PSI5_CH_CONFIG] bit if PCCR[PSI5_CH_CONFIG] is set for that channel; otherwise, that particular channel continues to stay in Disable mode. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 1, then the reset value is "1"(PSI5 module is disabled after reset). Note that when GLOBAL_DISABLE_REQ == 1 then all the clock enables (shown in Figure 1) from the IP go to a 0 (inactive) state. This bit is writable in all the modes.

Table continues on the next page...

1. This register is not present if the specific instance is configured as a slave. Master or slave configuration of the specific PSI5 instance decides if the operating mode (DISABLE/CONFIG/NORMAL) as well as the staggering of the channels can be controlled independently. If the instance is configured as a master, then it controls its own operating mode, staggering of all the channels in that instance and the operating mode and the staggering of any associated slave instance. If the instance is configured as a slave, then its operating mode and stagerring can only be controlled by another master instance. Please refer to the Device Configuration Chapter for instance specific details.

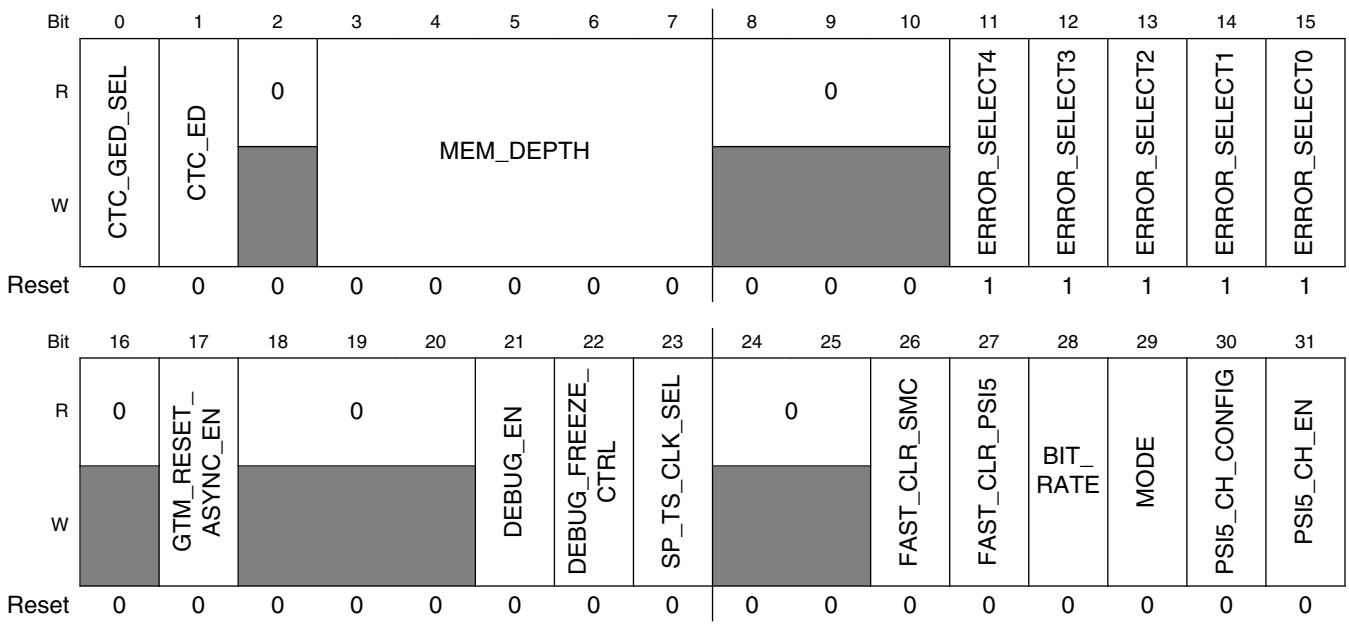
PSI5_GCR field descriptions (continued)

Field	Description
	0 PSI5 channel(s) enters Normal or Config mode, depending on the value of the PCCR[PSI5_CH_CONFIG] bit. 1 All PSI5 Channels are in Disable mode.

56.3.2 PSI5 Channel Control Register (PSI5_CH0_PCCR)

This section shows the PSI5 Channel Control Register.

Address: 0h base + 8h offset = 8h



PSI5_CH0_PCCR field descriptions

Field	Description
0 CTC_GED_SEL	Channel Target Counter, Global Enable/Disable Select. This bit selects whether the CTC should be enabled/disabled by CTC_GED bit (in GCR) or the CTC_ED bit. Enabling by the CTC_GED bit is used when staggering of the channels is required. 0 CTC enabled disabled by CTC_ED. 1 CTC enabled/disabled by CTC_GED.
1 CTC_ED	Channel Target Counter Enable/Disable. This bit is used to control the CTC locally from the channel in case the CTC_GED_SEL = 0. **NOTE:** This bit is writable in Normal mode and the Config mode. 0 The CTC counters is disabled and reset. 1 The CTC counter is enabled and start counting.

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 MEM_DEPTH	Can be programmed from 0 to 31 and denotes the size of the memory that should be used for storing the PSI5 messages. Area above the MEM_DEPTH is treated as being unavailable for message storage. MEM_DEPTH also denotes the number of RAM registers available for the CPU access. Note that the same memory can be accessed sequentially (though the DMA) or randomly anytime by the CPU. 0: it indicates that only 1 location of the memory is available for storage 31: it indicates that all the 32 locations of the memory are available for storage. Notes: • Depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR, the MEM_DEPTH memory size may be accessed by CPU only (interrupts are generated) or CPU and DMA (through the dedicated DMA popup registers). • DMA_PM_DS_CONFIG = conf1 then DMA requests are disabled and only the interrupts can be used for data transfer. • DMA_PM_DS_CONFIG = conf2,conf3,conf4 then DMA requests are enabled. The interrupts should be disabled by writing a 0 in the corresponding IE bits. • For more details please refer to description of DMA_PM_DS_CONFIG in the DCRs. • The data in the memory area defined by the MEM_DEPTH parameter is always filled in a sequential fashion as in a FIFO, but can be read sequentially or randomly. There can be no slot-memory location correspondence due to the T bit errors in the manchester decoder. • The memory area above the configured MEM_DEPTH size will always be read as 0. • Irrespective of the configuration, the data can always be randomly read by the CPU from the PMRL/PMRH register.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERROR_ SELECT4	The ERROR_SELECT bitfiield indicates which of the C, E, EM, T, or F error conditions generate an interrupt when the EICR Interrupt Enable bit is set. The individual bits are mapped as {C,E,EM,T,F}. The corresponding error is also latched in the PSI5_EISR. Default: ERROR_SELECT = 11111: Any of C,E,EM,T,or F generate an interrupt. C error CRC Error (C) Bit 20: ERROR_SELECT[4] or C_INT_SEL 0 The C bit does not generate an interrupt. 1 The C bit generates an interrupt.
12 ERROR_ SELECT3	Error_Select[3] or E_INT_SEL E error Electrical Error (E) 0 The E bit does not generate an interrupt. 1 The E bit generates an interrupt.
13 ERROR_ SELECT2	Error_Select[2] or EM_INT_SEL

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	0 The EM bit does not generate an interrupt. 1 The EM bit generates an interrupt.
14 ERROR_ SELECT1	Error_Select[1] or T_INT_SEL 0 The T bit does not generate an interrupt. 1 The T bit generates an interrupt.
15 ERROR_ SELECT0	Error_Select[0] or F_INT_SEL 0 The F bit does not generate an interrupt. 1 The F bit generates an interrupt.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 GTM_RESET_ ASYNC_EN	GTM_RESET_ASYNC_EN 0 Both the assertion and the deassertion of the GTM reset are treated synchronously. Inside the IP the gtm reset is synchronized on the SP_TS_CLK. 1 Assertion of the GTM reset is treated asynchronously but deassertion is synchronized on the SP_TS_CLK.(For details of SP_TS_CLK, refer to bit 23 of) PSI5 Channel Control Register (PSI5_CH0_PCCR)
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 DEBUG_EN	This bit allows/prevents the IP from entering the debug mode, whenever the debugger is connected and a breakpoint is encountered by the debugger: 0 The IP never enters the debug mode, even if the debugger is connected and a breakpoint is encountered. It keeps on working normally, as in the functional mode. 1 Whenever the debugger is connected and a breakpoint is encountered, the IP enters the debug mode. In the Debug mode, the DMA read pointers and other status registers that normally get affected by the READ operation remain unaffected.
22 DEBUG_ FREEZE_CTRL	DEBUG_FREEZE_CTRL 0 When the IP enters the Debug mode then it goes into the “Open” mode, i.e., the core of the IP keeps on working normally. In this case, there can be a difference between the value of a particular status bit, which is there at the time Debug mode is entered and the time at which the debugger reads it, since some upcoming events might change the status. The hardware clearing, of course, remains disabled. **NOTE:** In the above configuration, the status of registers read through the debugger may be different when entering debug mode and when actually accessing registers at some later point of time. 1 As soon as debug mode is asserted, the state of the system registers that are affected by the hardware gets frozen to what ever it was existing at the point at which the Debug mode goes active.This can be useful for reading the state of the system at a particular point of time such that it is NOT affected by the next upcoming events. Note that Manchester decoder/Sync Pulse Generator/ Time Stamp counters keep on working normally but the registers (data/status/timestamp) affected by these are not updated. When the IP enters the “Debug mode” then it goes into the “Freeze Mode”. Note that whenever there is a request to enter the freeze mode then the IP enters this mode, ONLY when the Manchester decoder and the Sync Pulse Generator are stopped coherently. This means that before entering the “freeze mode”, if there is any pending reception that is ongoing, then the same is completed first. Simultaneously the “sdout” line of the IP should be at a low level. Once these two

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	conditions are satisfied, the IP enters the Debug Freeze mode. The Time Stamp counters are also halted. The fact that the freeze mode has been entered, can be checked by reading the GISR[IS_DB_FR] bit. Once the IP comes out of the freeze mode then this bit goes to 0. **NOTE:** In the above configuration, incoming packets could get lost while the Debug mode is active.
23 SP_TS_CLK_SEL	This bit controls which clock goes to the Sync Pulse and Time Stamp Generator Unit. **NOTE:** The above selection affects ONLY the clock for the Time Stamp Unit and Sync Pulse Generator. The 1 MHz clock to the rest of the logic like the Manchester decoder or the Slot Boundary Counters is still the 1 MHz from the common clock generator module. **NOTE:** The clock selected using this bit is called SP_TS_CLK. 0 The 1 MHz clock generated from the common clock generator module goes for clocking the Time Stamp Unit and sync pulse generator. 1 The clock generated from the GTM goes for clocking the Time Stamp Unit and Sync Pulse Generator.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FAST_CLR_SMC	This bit controls the clearing mechanism of the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits in the GISR. **NOTE:** This bit is effective only when DCR[DMA_EN_SF] = 1. When FAST_CLR_SMC = 1 then the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits are automatically cleared upon reading the DSFR message register, as per the slot[x] message read in from the DSFR. When reading directly from the SFR registers, this bit does not affect auto clearing. If FAST_CLR_SMC = 0 then the above bits have to be cleared by a w1c. 0 Fast clearing is disabled (clear when written to 1). 1 Fast clearing is enabled.
27 FAST_CLR_PSI5	This bit control the clearing mechanism of bits in NDSR(in conf2, conf3 and conf4 modes), EISR (in conf2, conf3 and conf4 modes), and OWSR(in conf2 and conf3 modes). **NOTE:** This bit is effective only when DMA_PM_DS_CONFIG = conf2, conf3, or conf4 else this bit has no effect and only software clearing is available. Further, though the OWSR is not directly read by the DMA as a single register, it is cleared in specific modes (when FAST_CLR_PSI5 == 1) as can be seen below. Following are the details in the various modes of the DCR[DMA_PM_DS_CONFIG] bits: • In conf1 mode: This bit has no effect. Reading the DDSR and the DPMR returns a 0. • In conf2 mode: In this case the diagnostic bits (EISR, NDSR) are cleared ONLY when the diagnostic registers appended with the corresponding PSI5 messages are completely read using the DPMR. During the intermediate operation (when the PSI5 message has been read but the diagnostic bits are still pending), the PSI5 message read from the DPMR will not clear these bits. Both of these registers are simultaneously cleared when the corresponding DMA operation is over. However, only those bits are cleared, the corresponding messages of which form the part of the DMA request being serviced. Note that the OWSR bits are cleared upon the corresponding PSI5 message read from the DPMR. Reading the DDSR returns a 0 in this mode. • In conf3 mode: The PSI5 diagnostic register bits (EISR,NDSR) and OWSR are automatically cleared when the corresponding PSI5 message is read from the DPMR. The specific register bits of EISR,NDSR and OWSR, corresponding to each message, are cleared sequentially one by one as the corresponding message is being read by the DMA. Note that the three bits (each of EISR, NDSR, and OWSR) corresesponding to a specific location are always cleared simultaneoulsy. Reading the DDSR returns a 0. When reading from the PMRH/PMRL registers, this bit does not have any effect on fast clearing.

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	• In conf4 mode: In this case the diagnostic bits (EISR, NDSR) are cleared as soon as these registers are read through the DDSR. These registers are cleared simultaneously in one go, as soon as the corresponding DMA request has been serviced. Only those bits of these registers are cleared, the messages of which form a part of the DMA request. Note that in this mode the reading of the PSI5 messages from the DMA is not available as the request is being used for reading the diagnostic registers. Reading the DPMR returns a 0. The PSI5 messages can be read only from the PMRH/ PMRL register. Hence the OWSR has to be cleared only by w1c. • In all the above modes, when reading from the PMRH/PMRL registers, the FAST_CLR_PSI5 does not have any effect on fast clearing. • Further, In all the above cases if FAST_CLR_PSI5 = 0 then the specific diagnostic bits and the OWSR have to be cleared by w1c. 1 Fast Clearing Enabled 0 Fast Clearing Disables (clear when written to 1)
28 BIT_RATE	Bit Rate This bit selects the receive message bit rate (T bit) for this particular PSI5 Channel, that is, it selects the 4 MHz driven clock in the common clock generator module. 0 125 Kbit/s bit rate selected (4 MHz clock). 1 Reserved
29 MODE	This bit selects the operating modes. 0 Asynchronous operating mode (Integrated sync pulse generator module is switched off, only RX active). Note that in this mode the sync pulses from the internal sync pulse generator or the GTM are not allowed to be propagated to the output path, which always remains at logic 0. There is no T-bit error in the Asynchronous mode. 1 Synchronous operating modes (Integrated sync pulse generator module is switched on, both TX and RX subblocks active).
30 PSI5_CH_ CONFIG	PSI5 Channel Config mode request. If this bit is set (and PCCR[PSI5_CH_EN] = '1')this particular psi5 channel enters Configuration mode from Disable mode. Default value is 0. User software must program all PSI5 channel parameters before clearing this bit. Once cleared, channel parameters in registers are locked for writing. The values set can not be changed until this bit is set again after entering the Disable mode **NOTE:** This bit can be written in all modes, but writing a 0 in the Normal mode does not change the operation mode of the device. For more details please see Figure 56-2 and Table 56-2 . Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written. 1 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') - PSI5 Channel enters Config mode. Various configuration registers can be programmed in this mode. The only registers NOT writable in the config mode but writable in the Normal mode are the output registers PSI5_DBR and the PSI5_DSR. 0 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') PSI5 Channel enters Normal mode. Channel parameter registers are locked for writing. This bit can be written in all the modes but writing a "0" in the Normal mode will not change the operation mode of the device.
31 PSI5_CH_EN	PSI5 Channel Enable. If this bit is cleared this particular psi5 channel continues to stay in Disable mode even if GCR[GLOBAL_DISABLE_REQ] = 0. If set, it enters Configuration mode from Disable mode (based on PCCR[PSI5_CH_CONFIG]). Default value is 0. All state machines are disabled. This bit can be written in all modes.

Table continues on the next page...

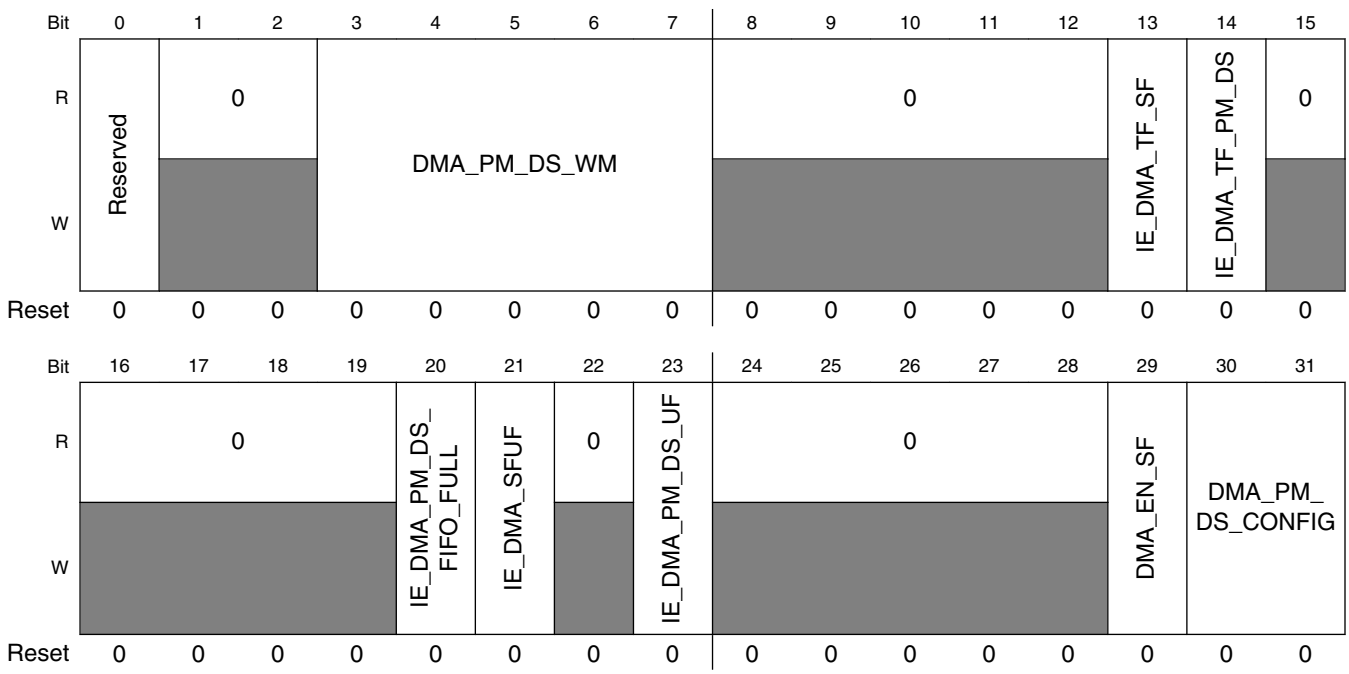
PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	NOTE: When PSI5_CH_EN == 0, then the clock enable for that particular channel that goes from the IP, goes to "0" (inactive state). This also means that the only clock available to the IP in the config mode is the "ipg_clk_s". The enable for the GTM Clock is not controlled by the IP and should be controlled externally. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written. 0 PSI5 channel continues in Disable mode. 1 PSI5 channel enabled to enter Config/Normal mode.

56.3.3 DMA Control Register (PSI5_CH0_DCR)

This section defines the DMA Control Register.

Address: 0h base + Ch offset = Ch



PSI5_CH0_DCR field descriptions

Field	Description
0 Reserved	This field is reserved. **NOTE:** Only write this field with the reset value.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH0_DCR field descriptions (continued)

Field	Description
3–7 DMA_PM_DS_WM	0 to 31: Valid water mark levels. Value fixed by user software. Since each location has two 32-bit registers hence a total of sixty-four 32-bit words are present. The default and minimum value of dma_pm_ds_wm is '0', i.e. 1 message/Diagnostic bit of 1 location. The maximum value of dma_pm_ds_wm is 31, i.e. the actual watermark = 32 RAM locations. **NOTE:** • These bits are ineffective if DMA_PM_DS_CONFIG = conf1. • These bits indicates the number of PSI5 message to be stored in FIFO / number of unread diagnostic bits in the diagnostic registers with a correspondence to the messages in the FIFO, before the corresponding DMA request is asserted. The watermark definition changes depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR as mentioned below. • When DMA_PM_DS_CONFIG = conf2 or conf3 then the watermark refers to the number of new unread PSI5 messages that are stored in the FIFO before a DMA request is asserted. • In conf2 mode of DMA_PM_DS_CONFIG, the number of words to be transferred, set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value + 2 words (for the two 32-bit diagnostic registers). • In conf3, the number of words to be transferred set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value. • When DMA_PM_DS_CONFIG = conf4 then the watermark refers to the number of new unread diagnostic bits in the diagnostic register. In this configuration the PSI5 messages can only be read by the CPU, since the DMA request would cater to reading the diagnostic bits only. • In conf4, the number of words transferred in each DMA request = 2 x 32-bit words (for the two 32-bit diagnostic registers). The water mark however refers to the number of new unread diagnostic bits pertaining to each PSI5 message in the FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IE_DMA_TF_SF	**NOTE:** This bit is writeable in Config and Normal modes. Enable for interrupt, which is generated when DMA transfer finishes for DMA SMC Frame register. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_DMA_TF_PM_DS	Enable for Interrupt, which is generated when DMA transfer finishes for PSI5 messages/DMA Diagnostic Status register depending on the DMA_PM_DS_CONFIG bits of the DCR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IE_DMA_PM_DS_FIFO_FULL	This bit is effective only when the DMA_PM_DS_CONFIG = conf2,conf3 or conf4. It enables the FIFO FULL condition generation interrupt. For the cases which generate these FIFO FULL conditions please refer to the description of IS_DMA_PM_DS_FIFO_FULL register bit of the DSR. **NOTE:** This bit is writable in Config and Normal modes.
21 IE_DMA_SFUF	

Table continues on the next page...

PSI5_CH0_DCR field descriptions (continued)

Field	Description
	Enables interrupt when there is underflow in DMA SMC Frame register when DMA is enabled. It is set when the DSFR is read without a valid DMA request, i.e. it is empty. Default value is 0. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IE_DMA_PM_DS_UF	Enables interrupt when there is underflow in DMA PSI5 message register when DMA is enabled. Default value is 0. For the details as to when this underflow occurs, please refer to the details of IS_DMA_PM_DS_UF bit details of the DSR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DMA_EN_SF	Enable DMA request for SMC Frame Data 0 DMA for SMC frame is disabled. The six dedicated 1-1 slot corresponding SMC registers can only be read by the CPU through the SFR. Reading the DSFRs returns 0. 1 DMA for SMC frame is enabled. The six dedicated slot corresponding SMC registers can be read by reading the DSFR successively. In the DMA mode the six registers are read in a round robin fashion as explained in the DSFR details. Reading directly through the SFR registers is still available.
30–31 DMA_PM_DS_CONFIG	These bits define how the PSI5 messages are stored in the MEM_DEPTH memory area and the associated diagnostic bits (EISR, NDSR) are transferred. The DPMR, DDSR, and DSFR registers should be used for DMA access and the PMRH/PMRL, NDSR, EISR, and OWSR registers should be used for CPU access. **NOTE:** For more details please see the description of the DCR[DMA_PM_DS_WM] bits. When DMA_PM_DS_CONFIG bits = conf2 or conf3 and if the CPU and the DMA are simultaneously used for reading the PSI5 messages, then in the course of reading the messages it is possible that there can be an overflow, though some intermediate locations, which have been read by the CPU, are empty. This will occur because the DMA access is made independent of the CPU access. The DMA logic would never know that the CPU has already read a message which was scheduled to be read by the DMA. Whenever the DMA is used for reading the PSI5 message (conf2/conf3) modes then though the CPU can read the messages in parallel through the PMRH/PMRL registers, still for the purpose of setting the Overwrite/Overflow bits only a read from the DMA through the popup registers (DPMR) would be treated as valid read. This will also be true for the SMC messages when accessed simultaneously by the CPU/DMA. **NOTE:** These bits are writable only in config mode. 00 **conf1** The DMA request is disabled. In this mode, only the interrupts can be used for indicating that data transfer is required. 01 **conf2**

Table continues on the next page...

PSI5_CH0_DCR field descriptions (continued)

Field	Description
	After transferring the "dma_pm_ds_wm" number of PSI5 messages, there will be two additional 32-bit transfers in which the NDSR and the EISR registers are sequentially transferred through the DPMR registers by using DMA. 10 **conf3** "dma_pm_ds_wm" number of PSI5 messages is transferred in each DMA request. The DMA read is through the DPMR registers. 11 **conf4** Only the diagnostic bits are transferred through the DDSR. The DMA read is through the DDSR registers.

56.3.4 DMA Status Register (PSI5_CH0_DSR)

This section defines the DMA Status Register.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													IS_DMA_TF_SF	IS_DMA_TF_PM_DS	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				IS_DMA_PM_DS_FIFO_FULL	IS_DMA_SFUF	0	IS_DMA_PM_DS_UF	0							
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DSR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IS_DMA_TF_SF	This flag is set when DMA transfer finishes for DMA SMC Frame register.
14 IS_DMA_TF_PM_DS	This flag is set when DMA transfer finishes . For the various configurations of this request please refer to the descriptions of the DMA_PM_DS_CONFIG bits in the DCR. This flag is cleared by w1c.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IS_DMA_PM_DS_FIFO_FULL	Interrupt Status when there is FIFO FULL corresponding to DMA request. The FIFO FULL scenarios are detailed below with respect to the various settings of the DMA_PM_DS_CONFIG bit. **conf1:** FIFO FULL disabled **conf2 and conf3:** FIFO FULL occurs when all the memory locations become full in the PSI5 message FIFO. It indicates that without any read, any new upcoming message will now overwrite the existing messages and set the overwrite bits. This bit is cleared either by w1c or upon a PSI5 message read depending on the FAST_CLR_PSI5 bit. **conf4:** FIFO FULL occurs when there is a "REGISTER FULL" in the Diagnostic registers. FIFO FULL here means that the diagnostic registers contain 32 unread diagnostic bits. A new message starts changing the status of the diagnostic bits. In this configuration it is necessary to read the diagnostic bits through the DMA using the DDSR in order to prevent FIFO FULL. Of course the diagnostic registers NDSR/EISR can be read any time by the CPU, but to prevent FIFO FULL they HAVE to be read through the DDSR registers only. Also in this mode the PSI5 messages DONT contribute to the FIFO FULL. The PSI5 messages have to be read by the CPU using the PMRH/PMRL registers in order to prevent the overwrite bits from being set in the OWSR. In this mode the OWSR bits are NOT automatically cleared upon the PSI5 message read since the PSI5 message has to be read by the PMRH/PMRL register and NOT though the DPMR. The OWSR bits have to be cleared by a w1c. Reading the DPMR returns a 0 in this mode. This bit is cleared by a w1c. 0 No FIFO full. 1 FIFO full has occurred
21 IS_DMA_SFUF	SMC Frame DMA underflow: This happens when the DSFR has been read without a proper DMA request being asserted. The DSFR is empty and it is read. This bit is cleared by a w1c. 0 No underflow has occurred. 1 Underflow has occurred.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IS_DMA_PM_DS_UF	Depending on the DMA_PM_DS_CONFIG bits following is the underflow conditions: Default value is 0. **conf1:** underflow disabled **conf2 and conf3:** Underflow happens when the software reads the PSI5 message FIFO though the DPMR, beyond the available messages (empty area). **conf4:** Underflow happens when the diagnostic registers are read through the DDSR without a valid DMA request and the DDSR contains no new data. The bits are cleared by a w1c.

Table continues on the next page...

PSI5_CH0_DSR field descriptions (continued)

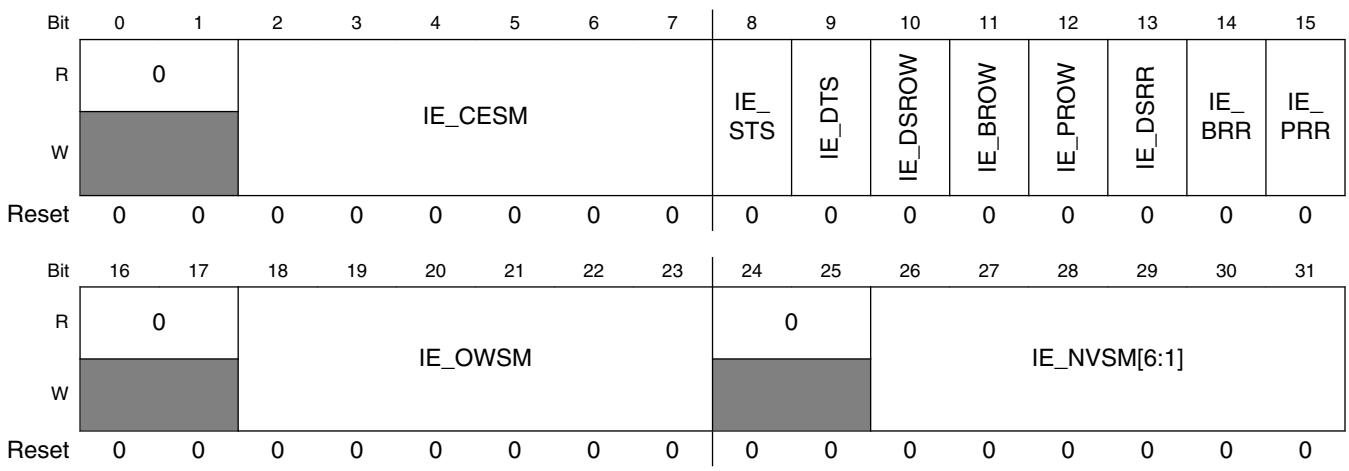
Field	Description
	0 No underflow has occurred. 1 Underflow has occurred.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.5 General Interrupt Control Register (PSI5_CH0_GICR)

The GICR contains the bits for controlling interrupts related to:

- SMC messages
- ECU-to-sensor communication
- the Time Stamp

Address: 0h base + 14h offset = 14h



PSI5_CH0_GICR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IE_CESM	Interrupt request when received SMC frame in corresponding slot has CRC failure (CRC recalculation on SMC). **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 IE_STS	Interrupt request enabled when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR. **NOTE:** This bit is writable in Config and Normal modes.

Table continues on the next page...

PSI5_CH0_GICR field descriptions (continued)

Field	Description
	0 Interrupt is disabled. 1 Interrupt is enabled.
9 IE_DTS	Interrupt request enabled when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_DSROW	Interrupt request enabled when system tries to overwrite on Data Shift register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 IE_BROW	Interrupt request enabled when system tries to overwrite on buffer register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_PROW	Interrupt request enabled when system tries to overwrite on Preparation register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_DSRR	Interrupt request enabled when Data Shift Register is ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_BRR	Interrupt request enabled when buffer Register is ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 IE_PRR	Interrupt request enabled when Preparation Register ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IE_OWSM	Interrupt request enable for the SMC message Overwrite bits.

Table continues on the next page...

PSI5_CH0_GICR field descriptions (continued)

Field	Description
	The interrupt for SMC message overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten the overwrite bit gets set. **NOTE:** These bits are writable in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IE_NVSM[6:1]	Interrupt request enabled when any new valid SMC message (fault free) is received in SFR[i]{i:0 to 5} i.e. during slot1 to slot6. This interrupt is generated only when corresponding IS_NVSMS[5:0] is set. **NOTE:** These bits are writable in Config and Normal modes. For each bit position, the corresponding request is enabled as follows: 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.6 New Data Interrupt Control Register (PSI5_CH0_NDICR)

The NDICR contains the interrupt enable bits related to a new PSI5 message arrival, in the corresponding PSI5 message buffer location.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ND																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_NDICR field descriptions

Field	Description
0–31 IE_ND	Interrupt request enabled when any new message (fault-free/with fault) is received in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.7 Overwrite Interrupt Control Register (PSI5_CH0_OWICR)

The Overwrite Interrupt Control Register (PSI5_OWICR) is related to the generation of the overwrite interrupts. The overwrite interrupt is generated when the unread PSI5 message in a particular RAM location is overwritten by another PSI5 message.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_OW																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_OWICR field descriptions

Field	Description
0–31 IE_OW	Interrupt request enabled when any new message overwrites the old unread PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt generated only when corresponding RAM buffer Ready with new data. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.8 Error Interrupt Control Register (PSI5_CH0_EICR)

The following figure shows the Error Interrupt Control Register.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ERROR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_EICR field descriptions

Field	Description
0–31 IE_ERROR	Interrupt request enabled when any/all of the error conditions C, E, EM, T, or F is observed in a PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when

PSI5_CH0_EICR field descriptions (continued)

Field	Description
	corresponding RAM buffer Ready with new data. Which of the error condition (any of C, E, EM, T, or F) would generate the interrupt is selectable by PCCR[ERROR_SELECT] field. If all are selected then which out of C,E,EM,T,F is set can be isolated by reading corresponding received message. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.9 General Interrupt Status Register (PSI5_CH0_GISR)

The GISR contains the status bits related to SMC messages, ECU-to-sensor communication, and the Time Stamp.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IS_DB_FR	0	IS_CESM						IS_STS	IS_DTS	IS_DSROW	IS_BROW	IS_PROW	DSR_RDY	DBR_RDY	DPR_RDY
W			w1c						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		IS_OWSM						0		IS_NVSM					
W			w1c								w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_GISR field descriptions

Field	Description
0 IS_DB_FR	This flag is set to “1” when the IP enters the Debug freeze mode. Please see Debug mode for details about the Debug mode. This bit is auto cleared by the hardware when the IP exits the debug freeze mode. 0 IP not in debug freeze mode. 1 IP in debug freeze mode.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IS_CESM	These flags are set when the received SMC frame in the corresponding slot has a CRC failure (CRC recalculation on SMC). These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.
8 IS_STS	This Interrupt flag is set when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR.
9 IS_DTS	This Interrupt flag is set when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR.
10 IS_DSROW	This flag is set when the system tries to overwrite on Data Shift register when it is not ready to accept new data.
11 IS_BROW	This flag is set when the system tries to overwrite on buffer register when it is not ready to accept new data.
12 IS_PROW	This flag is set when the system tries to overwrite on Preparation register when it is not ready to accept new data.
13 DSR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Shift register is ready to receive new data from the system bus side or the Data Buffer register. When "0" it indicates that there is a ongoing command transmission of the data in DSR , or the CPU has done a "w1c" to this bit leading to the new command transmission . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_DSROW] bit gets set. A write to PSI5_DOBCR[DSR_RST] will abort the current command transmission and make the DSR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, and PSI5_DOBCR[SW_READY] == 1 , then it indicates to the IP that the data in DSR is valid and is to be used for command shifting . This procedure is used when the data is written by the CPU to DSR directly skipping the DBR. For detailed action of this bit please refer to Data transmission . **NOTE:** For this field, Individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. **NOTE:** In configuration mode, writes to this bit are ignored.
14 DBR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Buffer register is ready to receive new data from the system bus side or the Data Preparation register. When "0" it indicates that there is a ongoing shift of the data from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_BROW] bit gets set. A write to PSI5_DOBCR[DBR_RST] will abort the current data transaction in DBR and make the DBR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DBR is valid and is to be used for shifting to DSR . This procedure is used when the data is written by the CPU to DBR directly skipping the DPR. For detailed action of this bit please refer to Data transmission . **NOTE:** For this field, individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used.

Table continues on the next page...

PSI5_CH0_GISR field descriptions (continued)

Field	Description
	NOTE: In configuration mode, writes to this bit are ignored.
15 DPR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Preparation register is ready to receive new data from the system bus side . When "0" it indicates that there is a ongoing processing of the data in DPR or there is a data shift ongoing from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_PROW] bit gets set. When PSI5_DOBCR[CMD_TYPE] == "7" then this bit remains at value "0". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DPR is valid and is to be used for processing and shifting to DBR . For detailed action of this bit please refer to Data transmission . **NOTE:** For these register bits individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. **NOTE:** In configuration mode, writes to this bit are ignored.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IS_OWSM	Status for the SMC message overwrite bits. Each of the 6 locations correspond to the SMC message pertaining to each SFR[x] register, x = 1 to 6. These bits are automatically cleared upon message read when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IS_NVSM	These flags are set when corresponding SFR[i] receives and is ready (to read) with new valid SMC message. These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.

56.3.10 DMA PSI5 Message Register (PSI5_CH0_DPMR)

This section defines the DMA PSI5 Message Register.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DPMR field descriptions

Field	Description
0–31 PSI5_RXDATA	PSI5_RXDATA[31:0]

PSI5_CH0_DPMR field descriptions (continued)

Field	Description
	This popup register contain the PMRL[31:0] data followed by the PMRH[31:0] data for each of the FIFO locations, when the FIFO is read by the DMA. It would require two 32-bit IPS access to read one PSI5 Message. For each of the locations the DATA is pushed sequentially till all the data corresponding to the programmed watermark is complete after which the DMA request goes down. The depth of the FIFO is 32x64. This popup register can be used differently based on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. When DMA_PM_DS_CONFIG=conf2 then it pops the PSI5 message followed by NDSR followed by EISR. When DMA_PM_DS_CONFIG= conf3, it pops only the PSI5 messages. When DMA_PM_DS_CONFIG=conf4 then it pops the NDSR followed by EISR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH0_PCCR) , DMA Control Register (PSI5_CH0_DCR) and DMA Status Register (PSI5_CH0_DSR) .

56.3.11 DMA SMC Frame Register (PSI5_CH0_DSFR)

This section shows the DMA SMC Frame Register.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SMC_RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DSFR field descriptions

Field	Description
0–31 SMC_RXDATA	When the DMA_EN_SF = 1 then the six SFR registers are searched in a round robin fashion for the reception of the complete SMC data. The DMA request is asserted as soon as the first encountered SFR has a complete SMC frame. This request remains asserted till all the registers which have a complete data ready have transferred the data through the DMA. Each other SFR is a 32-bit register. This register is a reflection of the location, pointed to by the DMA read pointer.

56.3.12 DMA Diagnostic Status Register (PSI5_CH0_DDSR)

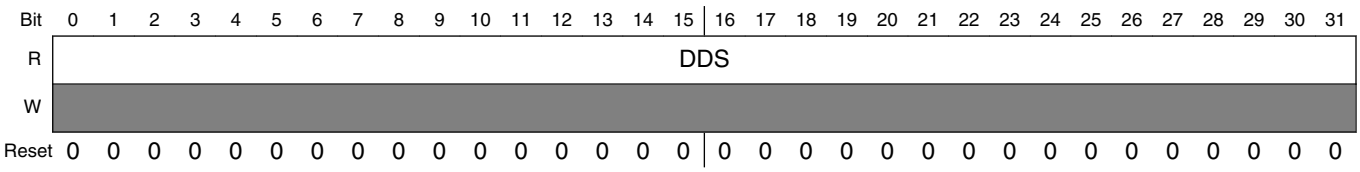
This section defines the DMA Diagnostic Status Register.

NOTE

No FIFO is implemented for diagnostic registers. They are two 32-bit registers addressable by either the CPU (reading NDSR

or EISR directly) or the DMA (reading the DDSR successively).

Address: 0h base + 30h offset = 30h



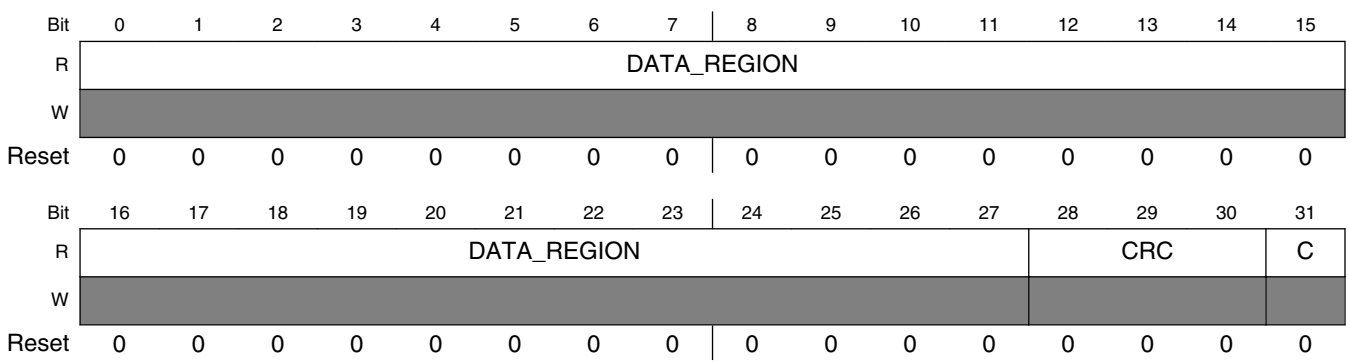
PSI5_CH0_DDSR field descriptions

Field	Description
0–31 DDS	This register maps each of the individual Diagnostic registers (in the order NDSR, EISR) depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH0_PCCR) , DMA Control Register (PSI5_CH0_DCR) and DMA Status Register (PSI5_CH0_DSR) .

56.3.13 PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)

This section defines the PSI5 Message Receive Register Low.

Address: 0h base + 34h offset = 34h



PSI5_CH0_PMRRL field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field.

Table continues on the next page...

PSI5_CH0_PMRRL field descriptions (continued)

Field	Description
28–30 CRC	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

56.3.14 PSI5 Message Receive Register High (PSI5_CH0_PMRRH)

This section defines the PSI5 Message Receive Register High.

NOTE

Any new PSI5 message is always stored in the PSI5 receive register (PMRRL/PMRRH). In parallel it is also stored in the RAM registers in a sequential manner like in a Ring Buffer. When a new message comes it would always overwrite the data in the PSI5 receive register. But the original message would still be available in the RAM register provided all locations in the RAM registers are not full, and an overwrite has not occured.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F	EM	E	T	SlotCounter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRRH field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 F	This represents that NO frame was received in the corresponding configured slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table continues on the next page...

PSI5_CH0_PMRRH field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0,M1), that is, at least one bit is absent during the T bit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This bit indicates electrical error, i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1, and start fields. 1 Electrical error in fields other than M0, M1, and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot, spread across two slots, started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 SlotCounter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,....Slot6.
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH0_S*n*FCR) . In other words these bits contain either the start bit(S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appendded to the message is "0" . It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM.

56.3.15 PSI5 Message Register Low i (PSI5_CH0_PMRL*n*)

The following figure shows the PSI5 Message Register Low i.

Address: 0h base + 3Ch offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_REGION															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_REGION												CRCP			C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRL*n* field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field. This bit is writeable in Config mode.
28–30 CRCP	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity. This bit is writeable in Config mode.
31 C	This bit will be set if CRC/P recalculation return an Error. This bit is writeable in Config mode.

56.3.16 PSI5 Message Register High i (PSI5_CH0_PMRH*n*)

The following figure shows the PSI5 Message Register High i.

Address: 0h base + 40h offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	O	F	EM	E	T	Slot_Counter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRH*n* field descriptions

Field	Description
0 O	This bit carries the same information as OWSR, but appended here for each message separately. This bit is set when the current message (which is being read) has overwritten some previous unread message. Note that, if the corresponding location in the OWSR has been cleared or has been set by the SOWSR, then the same will be reflected in the message stored in the corresponding RAM location as well.
1 F	This represents that NO frame was received in corresponding configured Slot. This bit is writable in Config mode (for Test purposes). 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table continues on the next page...

PSI5_CH0_PMRH*n* field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel(M0, M1) i.e. at least one bit is absent during the Tbit period. This bit is writable in Config mode (for Test purposes). 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This Bit Indicates Electrical Error i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). This bit is writable in Config mode (for Test purposes). 0 No electrical error in fields other than M0, M1 and start fields. 1 Electrical error in fields other than M0, M1 and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot or it has spread across two slots or it has started before the first configured slot. This bit is writable in Config mode (for Test purposes). 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 Slot_Counter	This value indicates the slot number in which this frame was recieved. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,....Slot6. This bit is writable in Config mode (for Test purposes).
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH0_S*n*FCR) . In other words these bits contain either the start bit (S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appendded to the message is "0". It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM. This bit is writable in Config mode (for Test purposes).

56.3.17 SMC Frame Register n (PSI5_CH0_SFR*n*)

The following figure shows the SMC Frame Register n.

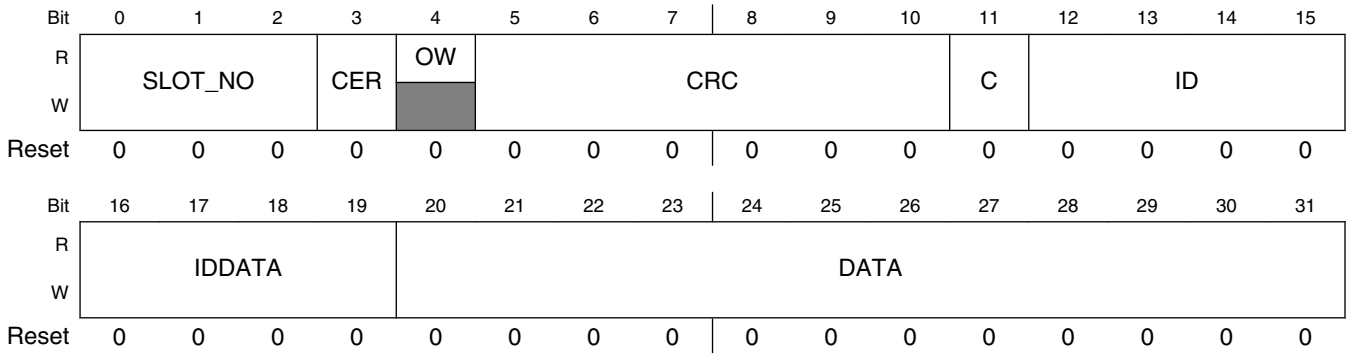
NOTE

Six dedicated 32-bit SFR[x] registers for SMC are available with a 1-1 slot correspondence. These can be read by either the CPU or the DMA (when DMA_EN_SF = 1).

NOTE

No FIFO for SMC frame is available. Six 32-bit registers are available in the address space, which can be accessed by either the DMA (using the DSFR) or the CPU (using the six SFR registers), depending on the setting of the DMA_EN_SMC bit.

Address: 0h base + 13Ch offset + (4d × i), where i=0d to 5d



PSI5_CH0_SFR*n* field descriptions

Field	Description
0–2 SLOT_NO	This indicates in which slot this SMC frame was received. Note that here the slot number refers to the slot in which all the re-arranged PSI5 messages (from which the M0,M1 bits have been extracted) are supposed to have occurred after the correct PSI5 message placement and Timing corrections have been performed. For SFR[1], slot_no = 1; For SFR[2], slot_no = 2; For SFR[3], slot_no = 3; For SFR[4], slot_no = 4; For SFR[5], slot_no = 5; For SFR[6], slot_no = 6; This field is writable in Config mode (for Test purposes).
3 CER	CRC Error This field is writable in Config mode (for Test purposes). 0 The present data has no CRC error. 1 The present data has a CRC error.
4 OW	Overwrite status **NOTE:** If the corresponding SOWSM bits in the SSESR have been set or the OWSM bits in the GISR have been cleared, then the same will be reflected in the message stored in the corresponding RAM location as well. The overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten, the overwrite bit gets set. 0 The present data was already read when new message arrived. 1 New message has overwritten the previous unread message.
5–10 CRC	6-bit CRC for slow serial message

Table continues on the next page...

PSI5_CH0_SFR*n* field descriptions (continued)

Field	Description
	This field is writable in Config mode (for Test purposes).
11 C	Configuration bit This bit is writable in Config mode (for Test purposes). 0 12-bit data and 8-bit message ID. 1 16-bit data and 4-bit message ID.
12–15 ID	Message ID: If C = '0' indicates ID[7:4], if C = '1' indicates ID[3:0]. This field is writable in Config mode (for Test purposes).
16–19 IDDATA	Message ID/DATA: If C = '0' indicates ID[3:0], if C = '1' indicates DATA[15:12]. This field is writable in Config mode (for Test purposes).
20–31 DATA	DATA payload This field is writable in Config mode (for Test purposes).

56.3.18 New Data Status Register (PSI5_CH0_NDSR)

The following figure shows the New Data Status Register.

Address: 0h base + 154h offset = 154h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NDS																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_NDSR field descriptions

Field	Description
0–31 NDS	New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when a new Message arrives i.e. faulty or non faulty in corresponding PSI5 MB location. Bit 0 reflects the MB0 location. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SNDR.

56.3.19 Overwrite Status Register (PSI5_CH0_OWSR)

Address: 0h base + 158h offset = 158h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OWS																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_OWSR field descriptions

Field	Description
0–31 OWS	Over Write Status flags for PSI5 Messages corresponding to each PSI5 location in the RAM registers. These bits are set when any unread receive PSI5 message is overwritten by a newly arrived message). These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SOWSR.

56.3.20 Error Indication Status Register (PSI5_CH0_EISR)

The following figure shows the Error Indication Status Register.

Address: 0h base + 15Ch offset = 15Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERROR																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_EISR field descriptions

Field	Description
0–31 ERROR	Error Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when any of the five selectable (by ERROR_SELECT) bits (C, E, EM, T or F) is set for the corresponding PSI5 Message. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SEISR.

56.3.21 Set New Data Status Register (PSI5_CH0_SNDSR)

The following figure shows the Set New Data Status Register.

Address: 0h base + 160h offset = 160h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SNDS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_SNDSR field descriptions

Field	Description
0–31 SNDS	Sets New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. The corresponding NDS Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** This bit is writable in config and Normal modes.

56.3.22 Set Overwrite Status Register (PSI5_CH0_SOWSR)

The following figure shows the Set Overwrite Status Register.

Address: 0h base + 164h offset = 164h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SOWS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

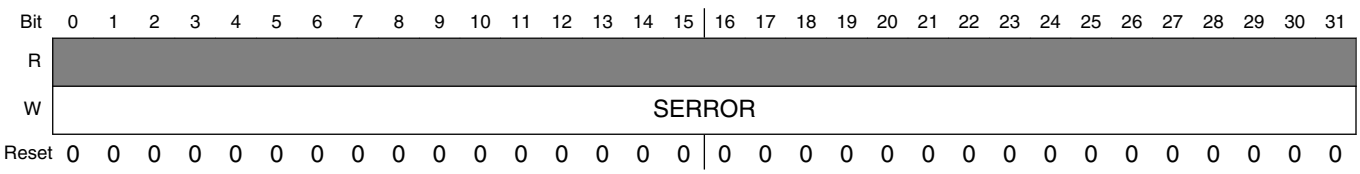
PSI5_CH0_SOWSR field descriptions

Field	Description
0–31 SOWS	Sets overwrite status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding OWS flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **NOTE:** These bits are writable in Config and Normal modes.

56.3.23 Set Error Status Register (PSI5_CH0_SEISR)

The following figure shows the Set Error Status Register.

Address: 0h base + 168h offset = 168h



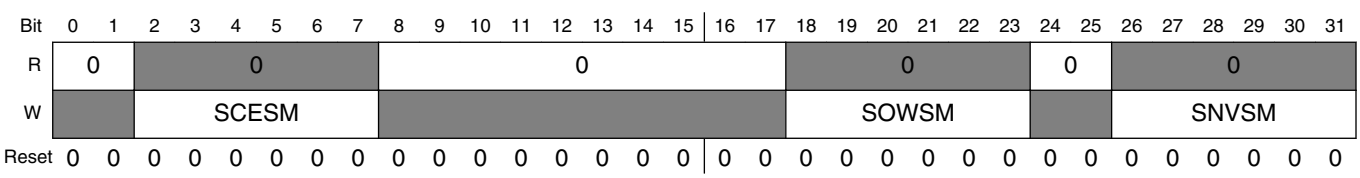
PSI5_CH0_SEISR field descriptions

Field	Description
0–31 SERROR	Sets Error Status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding ERROR flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **NOTE:** These bits are writable in Config and Normal modes.

56.3.24 Set SMC Error Status Register (PSI5_CH0_SSESR)

The following figure shows the Set SMC Error Status Register.

Address: 0h base + 16Ch offset = 16Ch



PSI5_CH0_SSESR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 SCESM	Sets IS_CESM Status flags for SMC Messages corresponding to each SMC MB locations.

Table continues on the next page...

PSI5_CH0_SSESR field descriptions (continued)

Field	Description
	The corresponding IS_CESM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** These bits are writable in Config and Normal modes.
8–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 SOWSM	Sets IS_OWSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_OWSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** These bits are writable in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SNVSM	Sets IS_NVSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_NVSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** This bit is writable in Config and Normal modes.

56.3.25 Sync Time Stamp Read Register (PSI5_CH0_STSRR)

The following figure shows the Sync Time Stamp Read Register.

Address: 0h base + 170h offset = 170h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								STSV																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_STSRR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 STSV	Sync Time Stamp Value[23:0] This bitfield provides the value of the free-running 24-bit time stamp counter triggered (captured) at last sync pulse.

56.3.26 Data Time Stamp Read Register (PSI5_CH0_DTSRR)

The following figure shows the Data Time Stamp Read Register.

NOTE

If S0 and S1 are detected in different slots then the Time Stamp value and the Slot Counter , both will correspond to the first slot(S0)

Address: 0h base + 174h offset = 174h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					SLOT_COUNTER			DTSV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DTSRR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SLOT_ COUNTER	3bit Slot Counter which corresponds to the slot for which the DTSRR contains the Time Stamp. The slot_counter contains the slot in which the start bits of this frame was captured (after successful detection of S1). If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. In case of asynchronous mode the slot counter has a fixed value of "1".
8–31 DTSV	Data Time Stamp Value[23:0] Whenever a valid sequence of the start bits (S0,S1) is detected then the value of the Time Stamp counter captured at the rising edge of the S0, is stored in this register at the rising edge of S1.

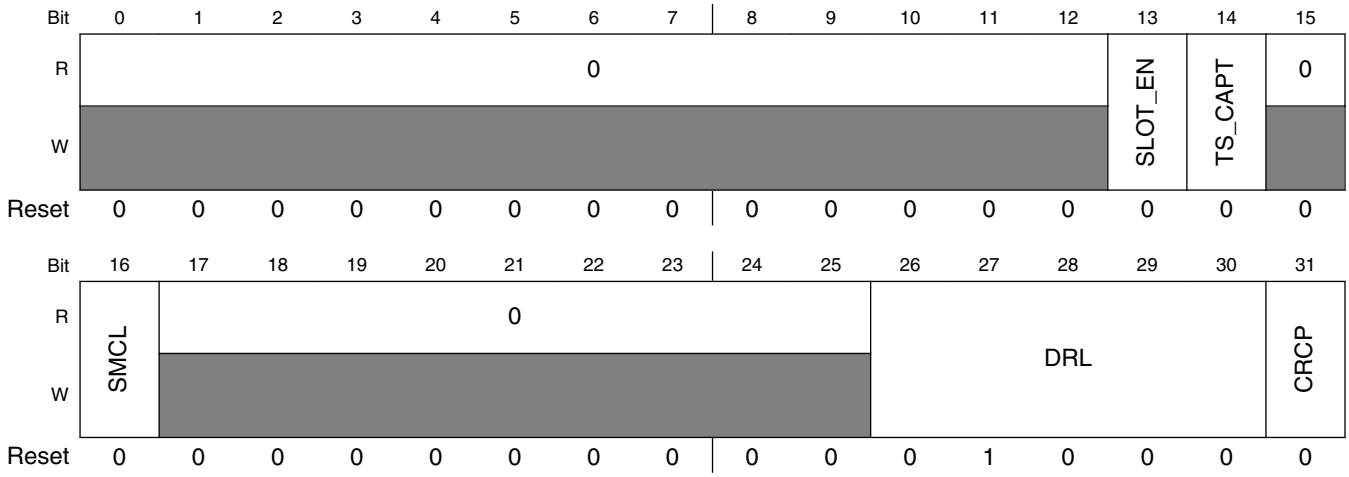
56.3.27 Slot n Frame Configuration Register (PSI5_CH0_S*n*FCR)

The following figure shows the Slot n Frame Configuration Register.

NOTE

In Asynchronous mode, the PSI5_S1FCR register has to be programmed for controlling the various parameters pertaining to the message received. The other registers PSI5_S[2-6]FCR have no effect in the asynchronous mode.

Address: 0h base + 178h offset + (4d × i), where i=0d to 5d



PSI5_CH0_S*n*FCR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SLOT_EN	SLOT_EN 0 This particular slot is disabled i.e not in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) have no effect. 1 This particular slot is enabled i.e is in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) are applicable.
14 TS_CAPT	TS_CAPT 0 Time stamp value captured at rising edge of S0 and stored Start sequence first edge. 1 Time stamp value captured at corresponding Tsync pulse (posedge) and stored with corresponding slot PSI5 message.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SMCL	Serial Messaging Channel field This field is used to enable the detection of start sequence of SMC frame on particular slot. Detection is not done if this field is '0'. 0 Serial Messaging Channel (optional) not present on Rx Message during Time Slot 'n' 1 2-bit{M1,M0} Serial Messaging Channel present on Rx Message during Time Slot 'n'
17–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–30 DRL	Data Region Length[4:0] This field represents Length of Data region (including Frame Control, Status , Data payload1&2 regions and optional messaging bits M0,M1) in Rx Message during Time Slot 'n' 00000-00111: Invalid 01000: 8 bit long data region 01001: 9 bit long data region 01010: 10 bit long data region ....

Table continues on the next page...

PSI5_CH0_S*n*FCR field descriptions (continued)

Field	Description
	11011: 27 bit long data region 11100: 28 bit long data region 11101-11111: Invalid
31 CRCP	CRCP 0 3 bit CRC[2:0] present on Rx Message during Time Slot 'n' 1 1 bit Parity field present on Rx Message during Time Slot 'n'

56.3.28 Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)

The following figure shows the Slot 1 Start Boundary Register.

Address: 0h base + 190h offset = 190h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S1SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S1SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S1SBT	Slot 1 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 1 should start.

56.3.29 Slot 2 Start Boundary Register (PSI5_CH0_S2SBR)

The following figure shows the Slot 2 Start Boundary Register.

Address: 0h base + 192h offset = 192h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S2SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S2SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S2SBT	Slot 2 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 2 should start.

56.3.30 Slot 3 Start Boundary Register (PSI5_CH0_S3SBR)

The following figure shows the Slot 3 Start Boundary Register.

Address: 0h base + 194h offset = 194h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S3SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S3SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S3SBT	Slot 3 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 3 should start.

56.3.31 Slot 4 Start Boundary Register (PSI5_CH0_S4SBR)

The following figure shows the Slot 4 Start Boundary Register.

Address: 0h base + 196h offset = 196h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S4SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S4SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S4SBT	Slot 4 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 4 should start.

56.3.32 Slot 5 Start Boundary Register (PSI5_CH0_S5SBR)

The following figure shows the Slot 5 Start Boundary Register.

Address: 0h base + 198h offset = 198h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S5SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S5SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S5SBT	Slot 5 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 5 should start.

56.3.33 Slot 6 Start Boundary Register (PSI5_CH0_S6SBR)

The following figure shows the Slot 6 Start Boundary Register.

Address: 0h base + 19Ah offset = 19Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S6SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S6SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S6SBT	Slot 6 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 6 should start.

56.3.34 Slot n End Boundary Register (PSI5_CH0_SnEBR)

The following figure shows the Slot *n* Start Boundary Register (PSI5_S *n* SBR).

Slot n End Boundary register is a single register which can be programmed to define the "end" of the nth slot. The n can be defined in the SLOT_NO field of this register.

There are seven boundary registers for the six slots. Six of these registers (S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR) are used to define the start of each of the six slots. The seventh register SnEBR (Slot n End Boundary register) defines the end of the nth slot. The n is programmed in the slot_no field of this register.

The number of slots would automatically be available from the SnEBR. The SnEBR contains the slot number of the last slot and the end boundary of the same. Each slot can separately be enabled or disabled by using the slot_en bit in the SnFCR. When the start slot boundaries are properly defined in the SnSBR (Slot n Start Boundary Register) register but the slot_en bit = 0 in the corresponding SnFCR (Slot n Frame Configuration register) register, then the corresponding slot is treated as "available" but "unconfigured." For such a slot the data/diagnostic bits all remain 0. The "T" bit will, however, get set if a message that was supposed to come in a configured slot actually arrives in an unconfigured slot. Note that the various parameters like the Data length/CRC configuration and so on are valid ONLY for the configured slots. Even when sampling a message in an unconfigured slot, these parameters would be taken from the previous/next configured slot only. It is assumed that no data would be sent in an unconfigured slot. If a data is received in this unconfigured slot it would never be treated as belonging to the unconfigured slot. As a special case if "S0" and "S1" are detected in different slots then a T-bit error is set .

NOTE

The start slot boundaries in the S *n* SBR should be correctly programmed. If this is not done, the correct operation cannot be guaranteed. Slots occurring after the slot_no value programmed in the SnEBR are treated as being unavailable (that is, the data

in all the slots occurring after the SnEBR is rejected by the Manchester decoder).

Following are the examples:

- If only three slots are required between two sync pulses, program S1SBR, S2SBR, and S3SBR with the correct value of the start boundaries of the three slots and program slot_no = 3 in the S *n* EBR and define the end boundary of the third slot in S *n* EBR. The design ignores slots 4, 5, and 6.
- For six slots, the required programming would be: program the start of each of the six slots in S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Then program the end of the sixth slot in the S *n* EBR by putting slot_no = 6 and define the end boundary of the sixth slot in S *n* EBR.

Address: 0h base + 19Ch offset = 19Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													SLOT_NO		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SnEBT														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_SnEBR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 SLOT_NO	The field indicates the Slot number{1-6} for which Slot End Boundary Time is defined. Any value other than 1-6 is defaulted to 0. A value 0 here would mean that the manchester decoder would not sample any slots. This slot number also defines the number of slots that the IP assumes to exist between 2 sync pulses.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 SnEBT	Slot *n* End Boundary Time. This field specifies the time in intervals of 1µs, occurring after the rising edge of timing sync pulse, at which the slot 'n" should end.

56.3.35 Data Output Block Configuration Register (PSI5_CH0_DOBCR)

The following figure shows the Data Output Block Configuration Register.

Table 56-4. CMD_TYPE details

CMD_TYPE	PSI5 Frame [1]	Effect on registers	Writeable Register lengths
"000" - 0	Short Frame(V1.3) with 31 "1s" as the start condition	Command can be written to DPR for auto calculation of stuff/start/CRC bits. Can always be overwritten in DSR/DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	6 bits in DPR 15bits[2] in DBR/DSR.
"001" - 1	Short Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"010" - 2	Long Frame(V1.3) with 31 "1s" as the start condition	-do-	16bits in DPR. 29bits[2] in DBR/DSR.
"011" - 3	Long Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"100" - 4	X-Long Frame(V1.3) with 31 "1s" as the start condition	-do-	22bits in DPR. 37bits[2] in DBR/DSR
"101" - 5	X-Long Frame(V1.3) with 5 "0s" as the start condition.	-do-	-do-
"110" - 6	XX-Long (V2.0) -	-do-	24 bits in DPR. 43bits[2] in DBR/DSR
"111" - 7	Non Standard Length.	Command can only be written to DSR and DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	Programmable from 1 to 64 bits in DBR/DSR, depending on the value of PSI5_DOBCR[DATA_LENGTH] field.

1. The start condition(31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command.
2. This is also the length of the command that will be shifted out during the ECU to Sensor communication, regardless of how many "actual" bits are written to DBR/DSR.

Table 56-5. DOBCR bit configurations and the related output states

Output path states	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL
State1	0	1	1
State2[1]	0	0	1
State3[1]	1	1	1
State4[1]	1	0	1
State5[1]	1	x	0

1. These states are shown in Valid states for integrated sync pulse generator. In State5, the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the "sdout" pin.

NOTE

Please refer to Figure 56-26 for the details about the above bits. Figure 56-26 is the actual implementation of Figure 56-25 such

that various contentions between the switches (S1, S2, and so on) do not happen.

Address: 0h base + 1A0h offset = 1A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA_LENGTH[5:0]						DBR_RST	DSR_RST	CMD_TYPE			DEFAULT_SYNC	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL	SW_READY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DOBCR field descriptions

Field	Description
0–5 DATA_LENGTH[5:0]	Can take on values from 0 to 63 corresponding to 1bit to 64bit non standard length commands. When PSI5_DOBCR[CMD_TYPE] == 7(Non Standard) , then this field controls the length of command that can be written to the DBR and the DSR registers, else the value in this field is ignored. This when DATA_LENGTH = 0, then the length of the command that can be written to DSR and the DBR is 1, when DATA_LENGTH = 1, then the length of the command that can be written to DSR and the DBR is 2 and so on, until a maximum of 64 bit length command when DATA_LENGTH = 63. Note that the length of the command that is programmed by using this field, is also equal to the length of the command that shifts out of the DSR register. In other words, it is equal to the number of bits that shift out of DSR register once the DSR_RDY goes low . After these many shifts have happened , then the value corresponding to "DEFAULT_SYNC" starts getting shifted out and DSR_RDY goes as "1".
6 DBR_RST	This is to reset and reject current content to Data Buffer Register. When this bit is written as "1" then the contents of the DBR are reset to all "0s" (if DEFAULT_SYNC = = 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DBR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
7 DSR_RST	This is to reset and reject current content to Data Shift Register.When this bit is written as "1" then the contents of the DSR are reset to all "0s" (if DEFAULT_SYNC = = 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DSR would be ready for new data.Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
8–10 CMD_TYPE	These 3 bits indicate the type of command that needs to be transmitted during the ECU to sensor communication. Table 56-4 is a brief description of the same.
11 DEFAULT_SYNC	When this bit is set to "0" then the default value of DSR and DBR registers are all "0s"; when this bit is set to 1 then the default value of DBR and DSR registers are all "1s". This value is used as the value of the default sync pulses shifted at the output path when the DSR_RDY == 1.
12 GTM_TRIG_SEL	GTM event triggered/internal sync pulse generator selection as shift clock for the DSR. 0 Internal sync pulse generator shift triggered 1 GTM event shift triggered
13 SP_PULSE_SEL	Selects the source for the short pulse PWM module: 0 SP module gets the data from the DSR 1 SP module directly gets input from the GTM_event/internal sync pulse generator

Table continues on the next page...

PSI5_CH0_DOBCR field descriptions (continued)

Field	Description
14 OP_SEL	This bit selects as to which would be driving source of the "ipp_do_psi5_sdout" port. 0 The sync pulse generator select as per the Bit3 1 PWM output
15 SW_READY	When this bit is written to 1 the transfer from DBR to DSR automatically happens as soon as DSR_RDY becomes 1. When this bit is kept to 0 then the transfer from DBR to DSR will remain pending . Once the software makes this bit as "1" and DSR_RDY also goes as "1" the transfer to DSR takes place. However, note that when this bit is 0, and the DSR_RDY becomes 1, still the transfer to the DSR will NOT happen. Only when this bit is a 1 , will the transfer happen. This bit can be written in CONFIG and the Normal modes.

56.3.36 Manchestor Decoder Disable Offset (PSI5_CH0_MDDIS_OFF)

The Manchester Decoder Disable Offset register (PSI5_MDDIS_OFF) defines the time in intervals of 1 µs, for which the Manchester decoder remains disabled after the falling edge of the sync pulse. Using this register, this time can be programmed from 0 µs to 128 µs. This works on the 1 µs clock from the common clock generator module.

Address: 0h base + 1A2h offset = 1A2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0									MDDIS_OFF						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_MDDIS_OFF field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MDDIS_OFF	These 7 bits can be used to program the time for which the Manchester Decoder remains disabled. This offset is added AFTER the falling edge of the sync pulse. Thus the total time for which the manchester decoder remains disabled = TsyncH + MDDIS_OFF where TsyncH is the high time for the sync pulse . Thus TsyncH = Pulse_Width0 or Pulse_Width1 as the case maybe, depending on whether a "0" is being pulse modulated or a "1". Note that the Manchester Decoder will ALWAYS remain disabled during the high time of the Sync Pulse. Using this register it is possible to disable the Manchester Decoder BEYOND the Sync Pulse High Time as well. Figure 1032 shows the "MDDIS_OFF" definition.

56.3.37 Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D)

The following figure shows the Pulse Width for Data Bit 0 Register.

Address: 0h base + 1A4h offset = 1A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								Pulse_Width0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PW0D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 Pulse_Width0	This defines the width (in µs) of data value '0' to be send from Data Output Register. It is the number of clock cycles(1µs clock) counted upto width "Pulse_Width0", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. When using PSI5 version 1.3 ECU-to-sensor Communication Pulse_Width0 should be configured to 0. Figure 1032 shows the definition of "Pulse_Width0".

56.3.38 Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D)

The following figure shows the Pulse Width for Data Bit 1 Register.

NOTE

When using the Version 1.3 format for ECU-to-sensor communication, the pulse width is NOT disabled. In this case, PW0D has to be programmed as 0. PW1D has to have the value of the length of the pulse desired when a 1 has to be transferred.

Address: 0h base + 1A6h offset = 1A6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								Pulse_Width1							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PW1D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH0_PW1D field descriptions (continued)

Field	Description
9–15 Pulse_Width1	This defines the width (in µs) of data value '1' to be send from Data Output Register. It is the number of clock cycles(1µs clock) counted upto width "Pulse_Width1", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. Figure 1032 shows the definition of "Pulse_Width1".

56.3.39 Counter Target Pulse Register (PSI5_CH0_CTPR)

The following figure shows the Counter Target Pulse Register.

NOTE

Please see Internal SYNC Pulse generation and coordination across PSI5 channels for details about the Sync Pulse generation and coordination.

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CTPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_CTPR field descriptions

Field	Description
0–15 CTPR	Counter Target Pulse Register. This is the target counter value. Once the CTC (channel target counter) reaches the value of CTPR this CTC is reset. CTC is a 16 bit free running counter that starts/stops depending on the CTC_GED (in GCR) bit or the CTC_ED bit (in PCCR). It gets reset initially when its value reaches CIPR and subsequently when its value reaches CTPR. The fact that which out of CTC_GED or the CTC_ED bit enables/disables it, is dependent on the CTC_GED_SEL bit in the PCCR. When CTC_GED/CTC_ED == 0 then the CTC is reset. The minimum value of CTPR is 6. Values less than 6 will be defaulted to 0 and the CTPR counter will not start. The value in CTPR indicates the Time Period of the internally generated sync pulses. The un modulated (No PWM) internally generated sync pulses have a high time of 4-sp_ts_clk cycles i.e. the duty cycle of the un modulated internally generated sync pulses is (4/|CTPR| *100). The 4 sp_ts_clk cycles is just the nascent value of the Pulse which actually triggers the PWD counters Note that this gives one more possible state, in addition to the states shown in Valid states for integrated sync pulse generator . This additional state allows the output of the internal sync pulse generator to be made directly available at the output, without being pulse modulated. The CTPR describes ONLY the SYNC pulse period and not the sync pulse length which is governed by the PWD registers, except in State5 where in the GTM governs these parameters.

56.3.40 Counter Initialize Pulse Register (PSI5_CH0_CIPR)

The following figure shows the Counter Initialize Pulse Register.

NOTE

Due to internal Flop synchronisation there can be a jitter of +/- 20 ns between the offset times that are programmed in the CIPR of different channels and the ones actually being observed. Similarly there can be a jitter of +/- 20 ns in the programmed CTPR and the measured CTPR.

NOTE

Please see Internal SYNC Pulse generation and coordination across PSI5 channels for details about the Sync Pulse generation and coordination.

Address: 0h base + 1AAh offset = 1AAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CIPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_CIPR field descriptions

Field	Description
0–15 CIPR	Counter initialize pulse register. This is the value using which the CTC gets reset, the first time after which it is enabled using CTC_GED (in GCR) or the CTC_ED bit (in PCCR). All subsequent resets are done when the CTC value reaches CTPR. The CIPR is used to set the offset time between the start of sync pulses between two different PSI5 channels, when the CTC of both of these channels are started simultaneously using CTC_GED. The offset between any two PSI5 channels would be |CIPR_channel1 - CIPR_channel2| sp_ts_clk cycles. This if channel 1 has a CIPR = 0 while channel2 has CIPR = 10, then the offset would be 10 sp_ts_clk cycles.

56.3.41 Data Preparation Register Low (PSI5_CH0_DPRL)

The following figure shows the Data Preparation Register Low.

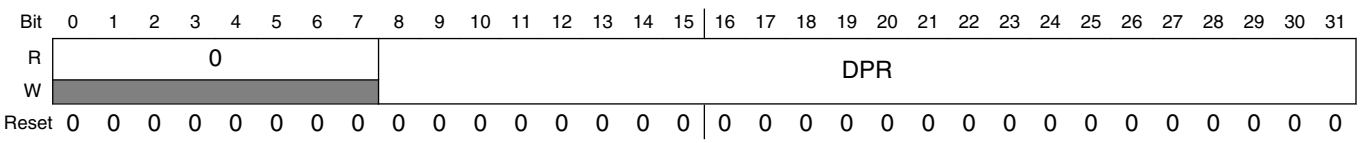
This register is the lower register used for writing the standard data and the address commands and is to be used for writing standard length ECU-to-sensor commands. The CRC/stuff/start bits are automatically appended by the hardware before the command is transferred to DBR. The commands to this register are accepted ONLY when PSI5_GISR[DPR_RDY] == 1. Whenever PSI5_GISR[DPR_RDY] == 1 , then each bit of

this register defaults to 0 . Writing to this register when PSI5_GISR[DPR_RDY] == 0 will cause the PSI5_GISR[IS_PROW] status bit to be set and the generation of an error interrupt, if it is enabled. Further , trying to write to this register when PSI5_DOBCR[CMD_TYPE] = "7" results in the setting of the PSI5_GISR[IS_PROW] bit and the resulting write is rejected.

Table 56-6. DPR details

PSI5_DOBCR[CMD_TYPE]	Standard PSI5 FRAME Types	Writeable bits in PSI5_DPRL
"0" or "1"	Short Frame(PSI5 V1.3)	6(DPR[5:0])
"2" or "3"	Long Frame(PSI5 V1.3)	16(DPR[15:0])
"4" or "5"	X-Long Frame(PSI5 V1.3)	22(DPR[21:0])
"6"	XX-Long Frame(PSI5 V2.0)	24(DPR[23:0])
"7"	Not Available for write	Not Available for write

Address: 0h base + 1ACh offset = 1ACh

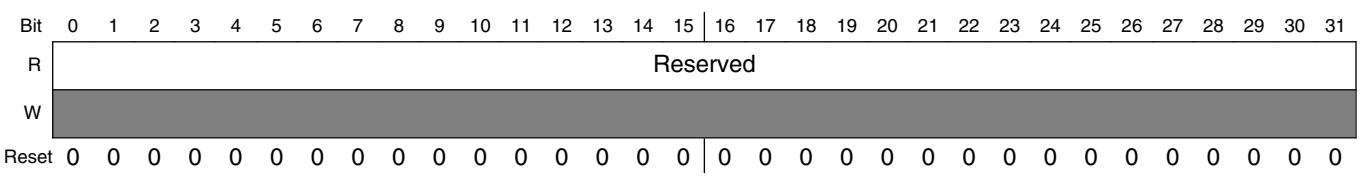


PSI5_CH0_DPRL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 DPR	DPR[23:0] are the 24-bits of the DPR register used for writing the variable length , standard ECU-to-Sensor commands , comprising of the address , data and other fields . Note that the IP is transparent to the arrangement of the address,data and other fields and it treats the bits in these fields identically for the purpose of CRC calculation and transmission. This register is writeable only when using the standard length PSI5 frames . For details about CRC calculation please refer to Data transmission . The writeable bits in this register are governed by the PSI5_DOBCR[CMD_TYPE] bit fields as per Table 56-6 . Note that the unwriteable bits in this register field are always read as "0s" . This register has to be written by the CPU. These bits can be written only in the Normal mode.

56.3.42 Data Preparation Register High (PSI5_CH0_DPRH)

Address: 0h base + 1B0h offset = 1B0h



PSI5_CH0_DPRH field descriptions

Field	Description
0–31 Reserved	This field is reserved. This register is always read as 0s. Writing to this register does not have any effect.

56.3.43 Data Buffer Register Low (PSI5_CH0_DBRL)

The following figure shows the Data Buffer Register Low.

This register is writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode.

NOTE

Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1, then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 56-7. DBRL[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" (Non Standard Length)	DBR[31:0]]	Variable length of writeable bits , as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
"6"(XX Long)	DBR[31:30]	DPR[19:18]
	DBR[29]	1'b0(stuff)
	DBR[28:23]	DPR[17:12]
	DBR[22]	1'b0(stuff)
	DBR[21:16	DPR[11:6]
	DBR[15]	1'b0(stuff)
	DBR[14:9]	DPR[5:0
	DBR[8:0]	"0111111110"

Table continues on the next page...

Table 56-7. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"4" or "5" (X-Long)	DBR[31]	1'b1(stuff)
	DBR[30:28]	DPR[20:18]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	DPR[17:15]
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"2" or "3" (Long)	DBR[31:29]	"000"
	DBR[28]	CRC[0]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	{CRC[1],CRC[2],DPR[15]}
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"0" or "1" (Short)	DBR[31:15]	0
	DBR[14:12]	{CRC[0],CRC[1],CRC[2]}[
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]

Table continues on the next page...

Table 56-7. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)

Address: 0h base + 1B4h offset = 1B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DBRL field descriptions

Field	Description
0–31 DBR	This register contains the Lower 32 bits(DBR[31:0]) of the max-64 bit length command (DBR[63:0]). The higher bits, DBR[63:32] are contained in the DBRH register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 56-7 . The table (except when PSI5_DOBCR[CMD_TYPE] = "7") show the bit assignment of the DBR[31:0] when these bits are automatically updated by the hardware. When PSI5_DOBCR[CMD_TYPE] == "7"), then only the CPU can write to these registers.

56.3.44 Data Buffer Register High (PSI5_CH0_DBRH)

The following figure shows the Data Buffer Register High.

This register is CPU writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY]== 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit

PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 56-8. DBRH[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" - Non Standard Frame lengths	DBR[63:32]]	Variable length of writeable bits , as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
"6" - XX Long	DBR[63:43]	All 0s
	DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1], CRC[0]}
	DBR[36]	1'b0(stuff)
	DBR[32:35]	DPR[23:20]
"4" or "5" -X Long	DBR[63:37]	All 0s
	DBR[36]	CRC[0]
	DBR[35]	1'b1((stuff)
	DBR[34:32]	{CRC[1],CRC[2],DPR[21]}
"3" or "2"(Long) OR "1" or "0" (Short)	DBR[63:32]	All 0s

Address: 0h base + 1B8h offset = 1B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DBRH field descriptions

Field	Description
0–31 DBR	This register contains the Upper 32 bits(DBR[63:32]) of the max-64 bit length command (DBR[63:0]). The lower bits, DBR[31:0] are contained in the DBRL register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 56-8 . Note that when PSI5_DOBCR[CMD_TYPE] == "7") , then only the CPU can write to these registers.

56.3.45 Data Shift Register Low (PSI5_CH0_DSRL)

The following figure shows the Data Shift Register Low.

This register is CPU writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addtion to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 1BCh offset = 1BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DSRL field descriptions

Field	Description
0–31 DSR	This register contains the lower 32 bits, (DSR[31:0]), of the max-64 bit length command (DSR[63:0]). The higher bits, DSR[63:32], are contained in the DSRH register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRL. Thus, when updated by the hardware, then DSR[31:0] = DBR[31:0]. Hardware updation occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0, then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1. These bits can be written by the CPU in the Normal mode , when PSI5_GISR[DSR_RDY] == 1'b1.

56.3.46 Data Shift Register High (PSI5_CH0_DSRH)

The following figure shows the Data Shift Register High.

This register is writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addtion to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 1C0h offset = 1C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

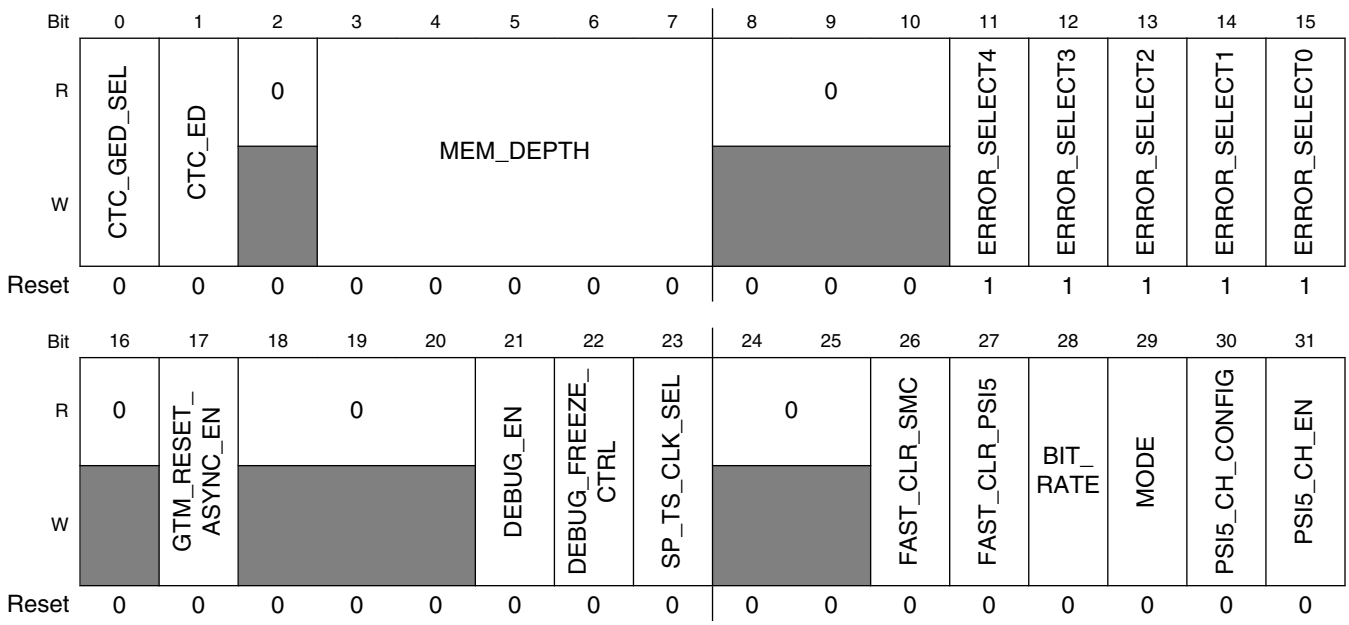
PSI5_CH0_DSRH field descriptions

Field	Description
0–31 DSR	This register contains the higher 32 bits (DSR[63:32) of the max-64 bit length command (DSR[63:0]). The lower bits, DSR[31:0], are contained in the DSRL register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRH. Thus when updated by the hardware, then DSR[63:32]] = DBR[63:32]. Hardware updation occurs when PSI5_GISR[DSR_RDY] == 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0, then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1 . These bits can be written by the CPU in the Normal mode when PSI5_GISR[DSR_RDY] == 1'b1 .

56.3.47 PSI5 Channel Control Register (PSI5_CH1_PCCR)

This section shows the PSI5 Channel Control Register.

Address: 0h base + 1C8h offset = 1C8h



PSI5_CH1_PCCR field descriptions

Field	Description
0 CTC_GED_SEL	Channel Target Counter, Global Enable/Disable Select. This bit selects whether the CTC should be enabled/disabled by CTC_GED bit (in GCR) or the CTC_ED bit. Enabling by the CTC_GED bit is used when staggering of the channels is required. 0 CTC enabled disabled by CTC_ED. 1 CTC enabled/disabled by CTC_GED.
1 CTC_ED	Channel Target Counter Enable/Disable. This bit is used to control the CTC locally from the channel in case the CTC_GED_SEL = 0. **NOTE:** This bit is writable in Normal mode and the Config mode. 0 The CTC counters is disabled and reset. 1 The CTC counter is enabled and start counting.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 MEM_DEPTH	Can be programmed from 0 to 31 and denotes the size of the memory that should be used for storing the PSI5 messages. Area above the MEM_DEPTH is treated as being unavailable for message storage. MEM_DEPTH also denotes the number of RAM registers available for the CPU access.

Table continues on the next page...

PSI5_CH1_PCCR field descriptions (continued)

Field	Description
	Note that the same memory can be accessed sequentially (though the DMA) or randomly anytime by the CPU. 0: it indicates that only 1 location of the memory is available for storage 31: it indicates that all the 32 locations of the memory are available for storage. Notes: • Depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR, the MEM_DEPTH memory size may be accessed by CPU only (interrupts are generated) or CPU and DMA (through the dedicated DMA popup registers). • DMA_PM_DS_CONFIG = conf1 then DMA requests are disabled and only the interrupts can be used for data transfer. • DMA_PM_DS_CONFIG = conf2,conf3,conf4 then DMA requests are enabled. The interrupts should be disabled by writing a 0 in the corresponding IE bits. • For more details please refer to description of DMA_PM_DS_CONFIG in the DCRs. • The data in the memory area defined by the MEM_DEPTH parameter is always filled in a sequential fashion as in a FIFO, but can be read sequentially or randomly. There can be no slot-memory location correspondence due to the T bit errors in the manchester decoder. • The memory area above the configured MEM_DEPTH size will always be read as 0. • Irrespective of the configuration, the data can always be randomly read by the CPU from the PMRL/PMRH register.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERROR_ SELECT4	The ERROR_SELECT bitfiield indicates which of the C, E, EM, T, or F error conditions generate an interrupt when the EICR Interrupt Enable bit is set. The individual bits are mapped as {C,E,EM,T,F}. The corresponding error is also latched in the PSI5_EISR. Default: ERROR_SELECT = 11111: Any of C,E,EM,T,or F generate an interrupt. C error CRC Error (C) Bit 20: ERROR_SELECT[4] or C_INT_SEL 0 The C bit does not generate an interrupt. 1 The C bit generates an interrupt.
12 ERROR_ SELECT3	Error_Select[3] or E_INT_SEL E error Electrical Error (E) 0 The E bit does not generate an interrupt. 1 The E bit generates an interrupt.
13 ERROR_ SELECT2	Error_Select[2] or EM_INT_SEL 0 The EM bit does not generate an interrupt. 1 The EM bit generates an interrupt.
14 ERROR_ SELECT1	Error_Select[1] or T_INT_SEL 0 The T bit does not generate an interrupt. 1 The T bit generates an interrupt.

Table continues on the next page...

PSI5_CH1_PCCR field descriptions (continued)

Field	Description
15 ERROR_ SELECT0	Error_Select[0] or F_INT_SEL 0 The F bit does not generate an interrupt. 1 The F bit generates an interrupt.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 GTM_RESET_ ASYNC_EN	GTM_RESET_ASYNC_EN 0 Both the assertion and the deassertion of the GTM reset are treated synchronously. Inside the IP the gtm reset is synchronized on the SP_TS_CLK. 1 Assertion of the GTM reset is treated asynchronously but deassertion is synchronized on the SP_TS_CLK.(For details of SP_TS_CLK, refer to bit 23 of) PSI5 Channel Control Register (PSI5_CH1_PCCR)
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 DEBUG_EN	This bit allows/prevents the IP from entering the debug mode, whenever the debugger is connected and a breakpoint is encountered by the debugger: 0 The IP never enters the debug mode, even if the debugger is connected and a breakpoint is encountered. It keeps on working normally, as in the functional mode. 1 Whenever the debugger is connected and a breakpoint is encountered, the IP enters the debug mode. In the Debug mode, the DMA read pointers and other status registers that normally get affected by the READ operation remain unaffected.
22 DEBUG_ FREEZE_CTRL	DEBUG_FREEZE_CTRL 0 When the IP enters the Debug mode then it goes into the “Open” mode, i.e., the core of the IP keeps on working normally. In this case, there can be a difference between the value of a particular status bit, which is there at the time Debug mode is entered and the time at which the debugger reads it, since some upcoming events might change the status. The hardware clearing, of course, remains disabled. **NOTE:** In the above configuration, the status of registers read through the debugger may be different when entering debug mode and when actually accessing registers at some later point of time. 1 As soon as debug mode is asserted, the state of the system registers that are affected by the hardware gets frozen to what ever it was existing at the point at which the Debug mode goes active.This can be useful for reading the state of the system at a particular point of time such that it is NOT affected by the next upcoming events. Note that Manchester decoder/Sync Pulse Generator/ Time Stamp counters keep on working normally but the registers (data/status/timestamp) affected by these are not updated. When the IP enters the “Debug mode” then it goes into the “Freeze Mode”. Note that whenever there is a request to enter the freeze mode then the IP enters this mode, ONLY when the Manchester decoder and the Sync Pulse Generator are stopped coherently. This means that before entering the “freeze mode”, if there is any pending reception that is ongoing, then the same is completed first. Simultaneously the “sdout” line of the IP should be at a low level. Once these two conditions are satisfied, the IP enters the Debug Freeze mode. The Time Stamp counters are also halted. The fact that the freeze mode has been entered, can be checked by reading the GISR[IS_DB_FR] bit. Once the IP comes out of the freeze mode then this bit goes to 0. **NOTE:** In the above configuration, incoming packets could get lost while the Debug mode is active.
23 SP_TS_CLK_ SEL	This bit controls which clock goes to the Sync Pulse and Time Stamp Generator Unit.

Table continues on the next page...

PSI5_CH1_PCCR field descriptions (continued)

Field	Description
	NOTE: The above selection affects ONLY the clock for the Time Stamp Unit and Sync Pulse Generator. The 1 MHz clock to the rest of the logic like the Manchester decoder or the Slot Boundary Counters is still the 1 MHz from the common clock generator module. **NOTE:** The clock selected using this bit is called SP_TS_CLK. 0 The 1 MHz clock generated from the common clock generator module goes for clocking the Time Stamp Unit and sync pulse generator. 1 The clock generated from the GTM goes for clocking the Time Stamp Unit and Sync Pulse Generator.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FAST_CLR_SMC	This bit controls the clearing mechanism of the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits in the GISR. **NOTE:** This bit is effective only when DCR[DMA_EN_SF] = 1. When FAST_CLR_SMC = 1 then the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits are automatically cleared upon reading the DSFR message register, as per the slot[x] message read in from the DSFR. When reading directly from the SFR registers, this bit does not affect auto clearing. If FAST_CLR_SMC = 0 then the above bits have to be cleared by a w1c. 0 Fast clearing is disabled (clear when written to 1). 1 Fast clearing is enabled.
27 FAST_CLR_PSI5	This bit control the clearing mechanism of bits in NDSR(in conf2, conf3 and conf4 modes), EISR (in conf2, conf3 and conf4 modes), and OWSR(in conf2 and conf3 modes). Following are the details in the various modes of the DCR[DMA_PM_DS_CONFIG] bits: • In conf1 mode: This bit has no effect. Reading the DDSR and the DPMR returns a 0. • In conf2 mode: In this case the diagnostic bits (EISR, NDSR) are cleared ONLY when the diagnostic registers appended with the corresponding PSI5 messages are completely read using the DPMR. During the intermediate operation (when the PSI5 message has been read but the diagnostic bits are still pending), the PSI5 message read from the DPMR will not clear these bits. Both of these registers are simultaneously cleared when the corresponding DMA operation is over. However, only those bits are cleared, the corresponding messages of which form the part of the DMA request being serviced. Note that the OWSR bits are cleared upon the corresponding PSI5 message read from the DPMR. Reading the DDSR returns a 0 in this mode. • In conf3 mode: The PSI5 diagnostic register bits (EISR,NDSR) and OWSR are automatically cleared when the corresponding PSI5 message is read from the DPMR. The specific register bits of EISR,NDSR and OWSR, corresponding to each message, are cleared sequentially one by one as the corresponding message is being read by the DMA. Note that the three bits (each of EISR, NDSR, and OWSR) corresesponding to a specific location are always cleared simultaneoulsy. Reading the DDSR returns a 0. When reading from the PMRH/PMRL registers, this bit does not have any effect on fast clearing. • In conf4 mode: In this case the diagnostic bits (EISR, NDSR) are cleared as soon as these registers are read through the DDSR. These registers are cleared simultaneously in one go, as soon as the corresponding DMA request has been serviced. Only those bits of these registers are cleared, the messages of which form a part of the DMA request. Note that in this mode the reading of the PSI5 messages from the DMA is not available as the request is being used for reading the diagnostic registers. Reading the DPMR returns a 0. The PSI5 messages can be read only from the PMRH/ PMRL register. Hence the OWSR has to be cleared only by w1c. • In all the above modes, when reading from the PMRH/PMRL registers, the FAST_CLR_PSI5 does not have any effect on fast clearing. • Further, In all the above cases if FAST_CLR_PSI5 = 0 then the specific diagnostic bits and the OWSR have to be cleared by w1c.

Table continues on the next page...

PSI5_CH1_PCCR field descriptions (continued)

Field	Description
	NOTE: This bit is effective only when DMA_PM_DS_CONFIG = conf2, conf3, or conf4 else this bit has no effect and only software clearing is available. Further, though the OWSR is not directly read by the DMA as a single register, it is cleared in specific modes (when FAST_CLR_PSI5 == 1) as can be seen below. 1 Fast Clearing Enabled 0 Fast Clearing Disables (clear when written to 1)
28 BIT_RATE	This bit selects the receive message bit rate (T bit) for this particular PSI5 Channel, that is, it selects one of the two driven clocks (4 MHz or 6.048 MHz) in the common clock generator module. 0 125 Kbit/s bit rate selected (4 MHz clock). 1 189 Kbit/s bit rate selected (6.048 MHz clock).
29 MODE	This bit selects the operating modes. 0 Asynchronous operating mode (Integrated sync pulse generator module is switched off, only RX active). Note that in this mode the sync pulses from the internal sync pulse generator or the GTM are not allowed to be propagated to the output path, which always remains at logic 0. There is no T-bit error in the Asynchronous mode. 1 Synchronous operating modes (Integrated sync pulse generator module is switched on, both TX and RX subblocks active).
30 PSI5_CH_ CONFIG	PSI5 Channel Config mode request. If this bit is set (and PCCR[PSI5_CH_EN] = '1')this particular psi5 channel enters Configuration mode from Disable mode. Default value is 0. User software must program all PSI5 channel parameters before clearing this bit. Once cleared, channel parameters in registers are locked for writing. The values set can not be changed until this bit is set again after entering the Disable mode **NOTE:** This bit can be written in all modes, but writing a 0 in the Normal mode does not change the operation mode of the device. For more details please see Figure 56-2 and Table 56-2 . Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written. 1 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') - PSI5 Channel enters Config mode. Various configuration registers can be programmed in this mode. The only registers NOT writable in the config mode but writable in the Normal mode are the output registers PSI5_DBR and the PSI5_DSR. 0 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') PSI5 Channel enters Normal mode. Channel parameter registers are locked for writing. This bit can be written in all the modes but writing a "0" in the Normal mode will not change the operation mode of the device.
31 PSI5_CH_EN	PSI5 Channel Enable. If this bit is cleared this particular psi5 channel continues to stay in Disable mode even if GCR[GLOBAL_DISABLE_REQ] = 0. If set, it enters Configuration mode from Disable mode (based on PCCR[PSI5_CH_CONFIG]). Default value is 0. All state machines are disabled. This bit can be written in all modes. **NOTE:** When PSI5_CH_EN == 0 then the clock enable for that particular channel that goes from the IP, goes to "0" (inactive state). This also means that the only clock available to the IP in the config mode is the "ipg_clk_s".The enable for the GTM Clock is not controlled by the IP and should be controlled externally. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written.

Table continues on the next page...

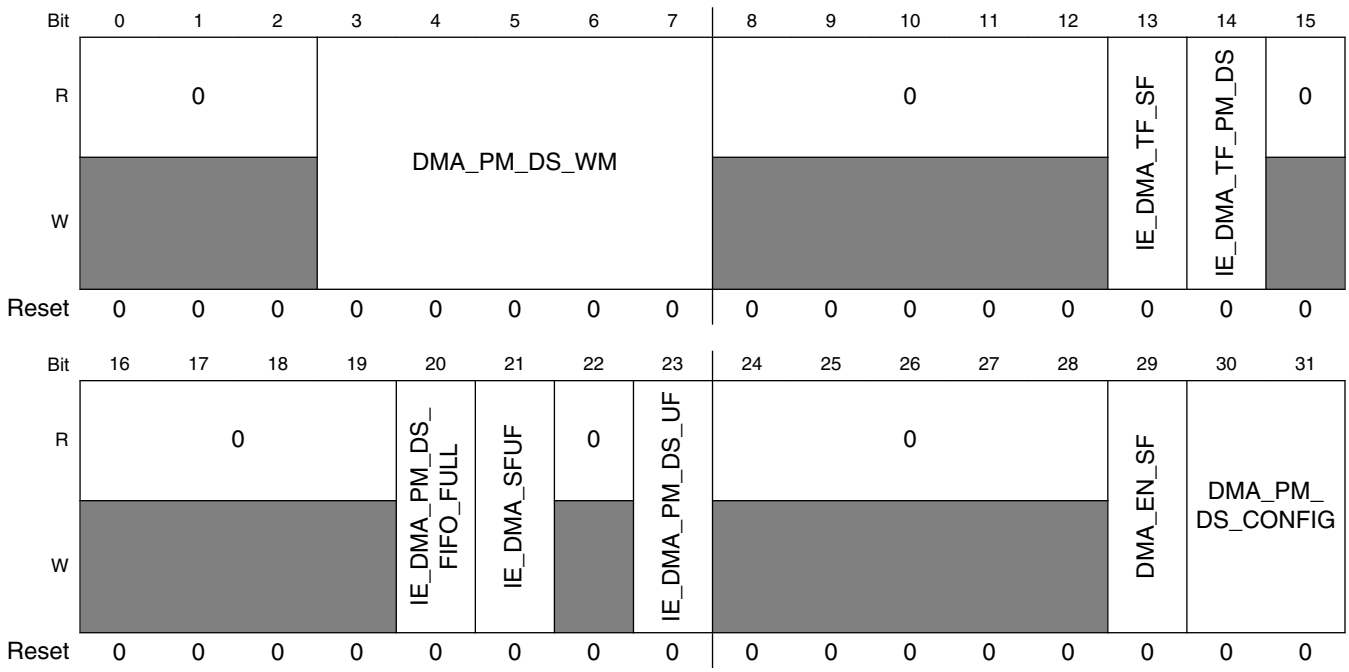
PSI5_CH1_PCCR field descriptions (continued)

Field	Description
	0 PSI5 channel continues in Disable mode. 1 PSI5 channel enabled to enter Config/Normal mode.

56.3.48 DMA Control Register (PSI5_CH1_DCR)

This section defines the DMA Control Register.

Address: 0h base + 1CCh offset = 1CCh



PSI5_CH1_DCR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 DMA_PM_DS_ WM	0 to 31: Valid water mark levels. Value fixed by user software. Since each location has two 32-bit registers hence a total of sixty-four 32-bit words are present. The default and minimum value of dma_pm_ds_wm is '0', i.e. 1 message/Diagnostic bit of 1 location. The maximum value of dma_pm_ds_wm is 31, i.e. the actual watermark = 32 RAM locations. **NOTE:** • These bits are ineffective if DMA_PM_DS_CONFIG = conf1. • These bits indicates the number of PSI5 message to be stored in FIFO / number of unread diagnostic bits in the diagnostic registers with a correspondence to the messages in the FIFO,

Table continues on the next page...

PSI5_CH1_DCR field descriptions (continued)

Field	Description
	before the corresponding DMA request is asserted. The watermark definition changes depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR as mentioned below. • When DMA_PM_DS_CONFIG = conf2 or conf3 then the watermark refers to the number of new unread PSI5 messages that are stored in the FIFO before a DMA request is asserted. • In conf2 mode of DMA_PM_DS_CONFIG, the number of words to be transferred, set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value + 2 words (for the two 32-bit diagnostic registers). • In conf3, the number of words to be transferred set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value. • When DMA_PM_DS_CONFIG = conf4 then the watermark refers to the number of new unread diagnostic bits in the diagnostic register. In this configuration the PSI5 messages can only be read by the CPU, since the DMA request would cater to reading the diagnostic bits only. • In conf4, the number of words transferred in each DMA request = 2 x 32-bit words (for the two 32-bit diagnostic registers). The water mark however refers to the number of new unread diagnostic bits pertaining to each PSI5 message in the FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IE_DMA_TF_SF	**NOTE:** This bit is writeable in Config and Normal modes. Enable for interrupt, which is generated when DMA transfer finishes for DMA SMC Frame register. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_DMA_TF_ PM_DS	Enable for Interrupt, which is generated when DMA transfer finishes for PSI5 messages/DMA Diagnostic Status register depending on the DMA_PM_DS_CONFIG bits of the DCR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IE_DMA_PM_ DS_FIFO_FULL	This bit is effective only when the DMA_PM_DS_CONFIG = conf2,conf3 or conf4. It enables the FIFO FULL condition generation interrupt. For the cases which generate these FIFO FULL conditions please refer to the description of IS_DMA_PM_DS_FIFO_FULL register bit of the DSR. **NOTE:** This bit is writable in Config and Normal modes.
21 IE_DMA_SFUF	Enables interrupt when there is underflow in DMA SMC Frame register when DMA is enabled. It is set when the DSFR is read without a valid DMA request, i.e. it is empty. Default value is 0. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH1_DCR field descriptions (continued)

Field	Description
23 IE_DMA_PM_DS_UF	Enables interrupt when there is underflow in DMA PSI5 message register when DMA is enabled. Default value is 0. For the details as to when this underflow occurs, please refer to the details of IS_DMA_PM_DS_UF bit details of the DSR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DMA_EN_SF	Enable DMA request for SMC Frame Data 0 DMA for SMC frame is disabled. The six dedicated 1-1 slot corresponding SMC registers can only be read by the CPU through the SFR. Reading the DSFRs returns 0. 1 DMA for SMC frame is enabled. The six dedicated slot corresponding SMC registers can be read by reading the DSFR successively. In the DMA mode the six registers are read in a round robin fashion as explained in the DSFR details. Reading directly through the SFR registers is still available.
30–31 DMA_PM_DS_CONFIG	These bits define how the PSI5 messages are stored in the MEM_DEPTH memory area and the associated diagnostic bits (EISR, NDSR) are transferred. The DPMR, DDSR, and DSFR registers should be used for DMA access and the PMRH/PMRL, NDSR, EISR, and OWSR registers should be used for CPU access. **NOTE:** For more details please see the description of the DCR[DMA_PM_DS_WM] bits. When DMA_PM_DS_CONFIG bits = conf2 or conf3 and if the CPU and the DMA are simultaneously used for reading the PSI5 messages, then in the course of reading the messages it is possible that there can be an overflow, though some intermediate locations, which have been read by the CPU, are empty. This will occur because the DMA access is made independent of the CPU access. The DMA logic would never know that the CPU has already read a message which was scheduled to be read by the DMA. Whenever the DMA is used for reading the PSI5 message (conf2/conf3) modes then though the CPU can read the messages in parallel through the PMRH/PMRL registers, still for the purpose of setting the Overwrite/Overflow bits only a read from the DMA through the popup registers (DPMR) would be treated as valid read. This will also be true for the SMC messages when accessed simultaneously by the CPU/DMA. **NOTE:** These bits are writable only in config mode. 00 **conf1** The DMA request is disabled. In this mode, only the interrupts can be used for indicating that data transfer is required. 01 **conf2** After transferring the "dma_pm_ds_wm" number of PSI5 messages, there will be two additional 32-bit transfers in which the NDSR and the EISR registers are sequentially transferred through the DPMR registers by using DMA. 10 **conf3** "dma_pm_ds_wm" number of PSI5 messages is transferred in each DMA request. The DMA read is through the DPMR registers. 11 **conf4** Only the diagnostic bits are transferred through the DDSR. The DMA read is through the DDSR registers.

56.3.49 DMA Status Register (PSI5_CH1_DSR)

This section defines the DMA Status Register.

Address: 0h base + 1D0h offset = 1D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													IS_DMA_TF_SF	IS_DMA_TF_PM_DS	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				IS_DMA_PM_DS_FIFO_FULL	IS_DMA_SFUF	0	IS_DMA_PM_DS_UF	0							
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_DSR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IS_DMA_TF_SF	This flag is set when DMA transfer finishes for DMA SMC Frame register.
14 IS_DMA_TF_PM_DS	This flag is set when DMA transfer finishes . For the various configurations of this request please refer to the descriptions of the DMA_PM_DS_CONFIG bits in the DCR. This flag is cleared by w1c.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IS_DMA_PM_DS_FIFO_FULL	Interrupt Status when there is FIFO FULL corresponding to DMA request. The FIFO FULL scenarios are detailed below with respect to the various settings of the DMA_PM_DS_CONFIG bit. **conf1:** FIFO FULL disabled **conf2 and conf3:** FIFO FULL occurs when all the memory locations become full in the PSI5 message FIFO. It indicates that without any read, any new upcoming message will now overwrite the existing messages and set the overwrite bits. This bit is cleared either by w1c or upon a PSI5 message read depending on the FAST_CLR_PSI5 bit. **conf4:** FIFO FULL occurs when there is a "REGISTER FULL" in the Diagnostic registers. FIFO FULL here means that the diagnostic registers contain 32 unread diagnostic bits. A new message starts changing the status of the diagnostic bits. In this configuration it is necessary to read the diagnostic bits through the DMA using the DDSR in order to prevent FIFO FULL. Of course the diagnostic registers NDSR/EISR can be read any time by the CPU, but to prevent FIFO FULL they HAVE to be read through the DDSR registers only. Also in this mode the PSI5 messages DONT contribute to the FIFO FULL. The PSI5 messages have to be read by the CPU using the PMRH/PMRL registers in order to prevent the overwrite bits from being set in the OWSR. In this mode the OWSR bits are NOT automatically cleared upon the PSI5 message read since the PSI5 message has to be read by the PMRH/PMRL register and NOT though the DPMR. The OWSR bits have to be cleared by a w1c. Reading the DPMR returns a 0 in this mode. This bit is cleared by a w1c. 0 No FIFO full. 1 FIFO full has occurred
21 IS_DMA_SFUF	SMC Frame DMA underflow: This happens when the DSFR has been read without a proper DMA request being asserted. The DSFR is empty and it is read. This bit is cleared by a w1c. 0 No underflow has occurred. 1 Underflow has occurred.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IS_DMA_PM_DS_UF	Depending on the DMA_PM_DS_CONFIG bits following is the underflow conditions: Default value is 0. **conf1:** underflow disabled **conf2 and conf3:** Underflow happens when the software reads the PSI5 message FIFO though the DPMR, beyond the available messages (empty area). **conf4:** Underflow happens when the diagnostic registers are read through the DDSR without a valid DMA request and the DDSR contains no new data. The bits are cleared by a w1c.

Table continues on the next page...

PSI5_CH1_DSR field descriptions (continued)

Field	Description
	0 No underflow has occurred. 1 Underflow has occurred.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.50 General Interrupt Control Register (PSI5_CH1_GICR)

The GICR contains the bits for controlling interrupts related to:

- SMC messages
- ECU-to-sensor communication
- the Time Stamp

Address: 0h base + 1D4h offset = 1D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		IE_CESM						IE_ STS	IE_DTS	IE_DSROW	IE_BROW	IE_PROW	IE_DSRR	IE_ BRR	IE_ PRR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		IE_OWSM						0		IE_NVSM[6:1]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_GICR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IE_CESM	Interrupt request when received SMC frame in corresponding slot has CRC failure (CRC recalculation on SMC). **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 IE_STS	Interrupt request enabled when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR. **NOTE:** This bit is writable in Config and Normal modes.

Table continues on the next page...

PSI5_CH1_GICR field descriptions (continued)

Field	Description
	0 Interrupt is disabled. 1 Interrupt is enabled.
9 IE_DTS	Interrupt request enabled when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_DSROW	Interrupt request enabled when system tries to overwrite on Data Shift register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 IE_BROW	Interrupt request enabled when system tries to overwrite on buffer register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_PROW	Interrupt request enabled when system tries to overwrite on Preparation register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_DSRR	Interrupt request enabled when Data Shift Register is ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_BRR	Interrupt request enabled when buffer Register is ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 IE_PRR	Interrupt request enabled when Preparation Register ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IE_OWSM	Interrupt request enable for the SMC message Overwrite bits.

Table continues on the next page...

PSI5_CH1_GICR field descriptions (continued)

Field	Description
	The interrupt for SMC message overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten the overwrite bit gets set. **NOTE:** These bits are writable in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IE_NVSM[6:1]	Interrupt request enabled when any new valid SMC message (fault free) is received in SFR[i]{i:0 to 5} i.e. during slot1 to slot6. This interrupt is generated only when corresponding IS_NVSMS[5:0] is set. **NOTE:** These bits are writable in Config and Normal modes. For each bit position, the corresponding request is enabled as follows: 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.51 New Data Interrupt Control Register (PSI5_CH1_NDICR)

The NDICR contains the interrupt enable bits related to a new PSI5 message arrival, in the corresponding PSI5 message buffer location.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 1D8h offset = 1D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ND																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_NDICR field descriptions

Field	Description
0–31 IE_ND	Interrupt request enabled when any new message (fault-free/with fault) is received in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.52 Overwrite Interrupt Control Register (PSI5_CH1_OWICR)

The Overwrite Interrupt Control Register (PSI5_OWICR) is related to the generation of the overwrite interrupts. The overwrite interrupt is generated when the unread PSI5 message in a particular RAM location is overwritten by another PSI5 message.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 1DCh offset = 1DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_OW																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_OWICR field descriptions

Field	Description
0–31 IE_OW	Interrupt request enabled when any new message overwrites the old unread PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt generated only when corresponding RAM buffer Ready with new data. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.53 Error Interrupt Control Register (PSI5_CH1_EICR)

The following figure shows the Error Interrupt Control Register.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 1E0h offset = 1E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ERROR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_EICR field descriptions

Field	Description
0–31 IE_ERROR	Interrupt request enabled when any/all of the error conditions C, E, EM, T, or F is observed in a PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when

PSI5_CH1_EICR field descriptions (continued)

Field	Description
	corresponding RAM buffer Ready with new data. Which of the error condition (any of C, E, EM, T, or F) would generate the interrupt is selectable by PCCR[ERROR_SELECT] field. If all are selected then which out of C,E,EM,T,F is set can be isolated by reading corresponding received message. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.54 General Interrupt Status Register (PSI5_CH1_GISR)

The GISR contains the status bits related to SMC messages, ECU-to-sensor communication, and the Time Stamp.

Address: 0h base + 1E4h offset = 1E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IS_DB_FR	0	IS_CESM						IS_STS	IS_DTS	IS_DSROW	IS_BROW	IS_PROW	DSR_RDY	DBR_RDY	DPR_RDY
W			w1c						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		IS_OWSM						0		IS_NVSM					
W			w1c								w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_GISR field descriptions

Field	Description
0 IS_DB_FR	This flag is set to “1” when the IP enters the Debug freeze mode. Please see Debug mode for details about the Debug mode. This bit is auto cleared by the hardware when the IP exits the debug freeze mode. 0 IP not in debug freeze mode. 1 IP in debug freeze mode.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IS_CESM	These flags are set when the received SMC frame in the corresponding slot has a CRC failure (CRC recalculation on SMC). These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.
8 IS_STS	This Interrupt flag is set when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR.
9 IS_DTS	This Interrupt flag is set when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR.
10 IS_DSROW	This flag is set when the system tries to overwrite on Data Shift register when it is not ready to accept new data.
11 IS_BROW	This flag is set when the system tries to overwrite on buffer register when it is not ready to accept new data.
12 IS_PROW	This flag is set when the system tries to overwrite on Preparation register when it is not ready to accept new data.
13 DSR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Shift register is ready to receive new data from the system bus side or the Data Buffer register. When "0" it indicates that there is a ongoing command transmission of the data in DSR , or the CPU has done a "w1c" to this bit leading to the new command transmission . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_DSROW] bit gets set. A write to PSI5_DOBCR[DSR_RST] will abort the current command transmission and make the DSR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, and PSI5_DOBCR[SW_READY] == 1 , then it indicates to the IP that the data in DSR is valid and is to be used for command shifting . This procedure is used when the data is written by the CPU to DSR directly skipping the DBR. For detailed action of this bit please refer to Data transmission . **NOTE:** For this field, Individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. **NOTE:** In configuration mode, writes to this bit are ignored.
14 DBR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Buffer register is ready to receive new data from the system bus side or the Data Preparation register. When "0" it indicates that there is a ongoing shift of the data from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_BROW] bit gets set. A write to PSI5_DOBCR[DBR_RST] will abort the current data transaction in DBR and make the DBR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DBR is valid and is to be used for shifting to DSR . This procedure is used when the data is written by the CPU to DBR directly skipping the DPR. For detailed action of this bit please refer to Data transmission . **NOTE:** For this field, individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used.

Table continues on the next page...

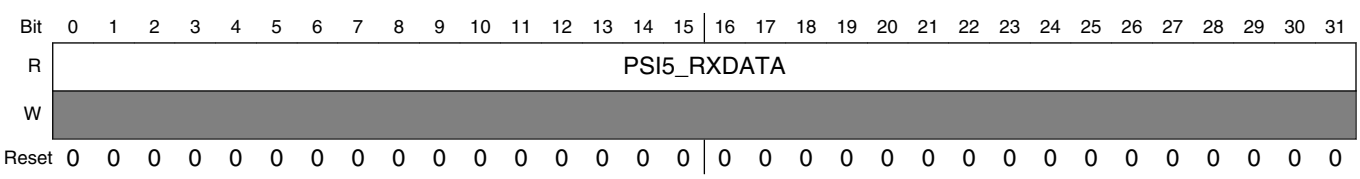
PSI5_CH1_GISR field descriptions (continued)

Field	Description
	NOTE: In configuration mode, writes to this bit are ignored.
15 DPR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Preparation register is ready to receive new data from the system bus side . When "0" it indicates that there is a ongoing processing of the data in DPR or there is a data shift ongoing from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_PROW] bit gets set. When PSI5_DOBCR[CMD_TYPE] == "7" then this bit remains at value "0". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DPR is valid and is to be used for processing and shifting to DBR . For detailed action of this bit please refer to Data transmission . **NOTE:** For these register bits individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. **NOTE:** In configuration mode, writes to this bit are ignored.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IS_OWSM	Status for the SMC message overwrite bits. Each of the 6 locations correspond to the SMC message pertaining to each SFR[x] register, x = 1 to 6. These bits are automatically cleared upon message read when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IS_NVSM	These flags are set when corresponding SFR[i] receives and is ready (to read) with new valid SMC message. These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.

56.3.55 DMA PSI5 Message Register (PSI5_CH1_DPMR)

This section defines the DMA PSI5 Message Register.

Address: 0h base + 1E8h offset = 1E8h



PSI5_CH1_DPMR field descriptions

Field	Description
0–31 PSI5_RXDATA	PSI5_RXDATA[31:0]

PSI5_CH1_DPMR field descriptions (continued)

Field	Description
	This popup register contain the PMRL[31:0] data followed by the PMRH[31:0] data for each of the FIFO locations, when the FIFO is read by the DMA. It would require two 32-bit IPS access to read one PSI5 Message. For each of the locations the DATA is pushed sequentially till all the data corresponding to the programmed watermark is complete after which the DMA request goes down. The depth of the FIFO is 32x64. This popup register can be used differently based on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. When DMA_PM_DS_CONFIG=conf2 then it pops the PSI5 message followed by NDSR followed by EISR. When DMA_PM_DS_CONFIG= conf3, it pops only the PSI5 messages. When DMA_PM_DS_CONFIG=conf4 then it pops the NDSR followed by EISR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH1_PCCR) , DMA Control Register (PSI5_CH1_DCR) and DMA Status Register (PSI5_CH1_DSR) .

56.3.56 DMA SMC Frame Register (PSI5_CH1_DSFR)

This section shows the DMA SMC Frame Register.

Address: 0h base + 1ECh offset = 1ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SMC_RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_DSFR field descriptions

Field	Description
0–31 SMC_RXDATA	When the DMA_EN_SF = 1 then the six SFR registers are searched in a round robin fashion for the reception of the complete SMC data. The DMA request is asserted as soon as the first encountered SFR has a complete SMC frame. This request remains asserted till all the registers which have a complete data ready have transferred the data through the DMA. Each other SFR is a 32-bit register. This register is a reflection of the location, pointed to by the DMA read pointer.

56.3.57 DMA Diagnostic Status Register (PSI5_CH1_DDSR)

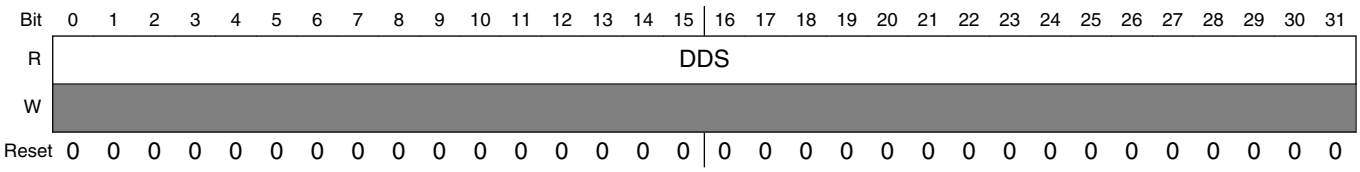
This section defines the DMA Diagnostic Status Register.

NOTE

No FIFO is implemented for diagnostic registers. They are two 32-bit registers addressable by either the CPU (reading NDSR

or EISR directly) or the DMA (reading the DDSR successively).

Address: 0h base + 1F0h offset = 1F0h



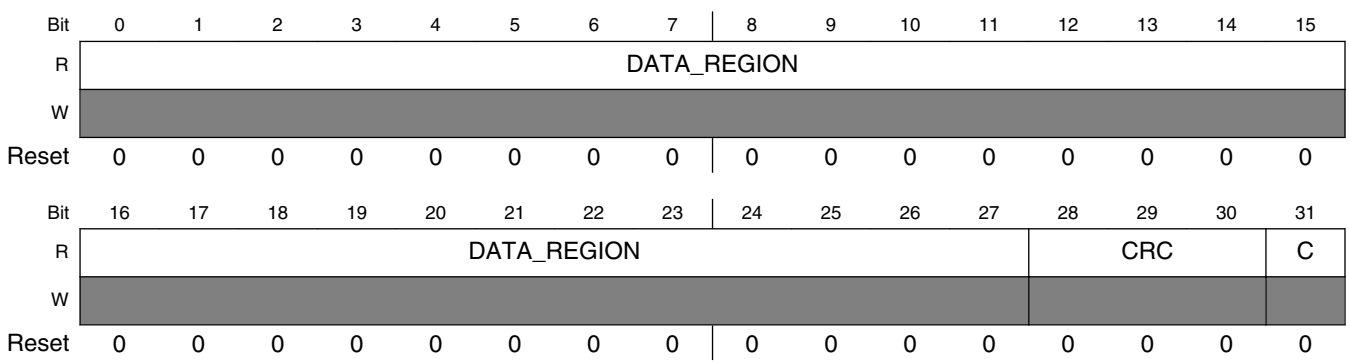
PSI5_CH1_DDSR field descriptions

Field	Description
0–31 DDS	This register maps each of the individual Diagnostic registers (in the order NDSR, EISR) depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH1_PCCR) , DMA Control Register (PSI5_CH1_DCR) and DMA Status Register (PSI5_CH1_DSR) .

56.3.58 PSI5 Message Receive Register Low (PSI5_CH1_PMRRL)

This section defines the PSI5 Message Receive Register Low.

Address: 0h base + 1F4h offset = 1F4h



PSI5_CH1_PMRRL field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field.

Table continues on the next page...

PSI5_CH1_PMRRL field descriptions (continued)

Field	Description
28–30 CRC	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

56.3.59 PSI5 Message Receive Register High (PSI5_CH1_PMRRH)

This section defines the PSI5 Message Receive Register High.

NOTE

Any new PSI5 message is always stored in the PSI5 receive register (PMRRL/PMRRH). In parallel it is also stored in the RAM registers in a sequential manner like in a Ring Buffer. When a new message comes it would always overwrite the data in the PSI5 receive register. But the original message would still be available in the RAM register provided all locations in the RAM registers are not full, and an overwrite has not occured.

Address: 0h base + 1F8h offset = 1F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F	EM	E	T	SlotCounter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_PMRRH field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 F	This represents that NO frame was received in the corresponding configured slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table continues on the next page...

PSI5_CH1_PMRRH field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0,M1), that is, at least one bit is absent during the T bit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This bit indicates electrical error, i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1, and start fields. 1 Electrical error in fields other than M0, M1, and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot, spread across two slots, started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 SlotCounter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,....Slot6.
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH1_S*n*FCR) . In other words these bits contain either the start bit(S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appendded to the message is "0" . It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM.

56.3.60 PSI5 Message Register Low i (PSI5_CH1_PMRL*n*)

The following figure shows the PSI5 Message Register Low i.

Address: 0h base + 1FCh offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_REGION															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_REGION												CRCP			C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_PMRL*n* field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field. This bit is writeable in Config mode.
28–30 CRCP	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity. This bit is writeable in Config mode.
31 C	This bit will be set if CRC/P recalculation return an Error. This bit is writeable in Config mode.

56.3.61 PSI5 Message Register High i (PSI5_CH1_PMRH*n*)

The following figure shows the PSI5 Message Register High i.

Address: 0h base + 200h offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	O	F	EM	E	T	Slot_Counter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_PMRH*n* field descriptions

Field	Description
0 O	This bit carries the same information as OWSR, but appended here for each message separately. This bit is set when the current message (which is being read) has overwritten some previous unread message. Note that, if the corresponding location in the OWSR has been cleared or has been set by the SOWSR, then the same will be reflected in the message stored in the corresponding RAM location as well.
1 F	This represents that NO frame was received in corresponding configured Slot. This bit is writable in Config mode (for Test purposes). 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table continues on the next page...

PSI5_CH1_PMRH*n* field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel(M0, M1) i.e. at least one bit is absent during the Tbit period. This bit is writable in Config mode (for Test purposes). 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This Bit Indicates Electrical Error i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). This bit is writable in Config mode (for Test purposes). 0 No electrical error in fields other than M0, M1 and start fields. 1 Electrical error in fields other than M0, M1 and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot or it has spread across two slots or it has started before the first configured slot. This bit is writable in Config mode (for Test purposes). 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 Slot_Counter	This value indicates the slot number in which this frame was recieved. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,....Slot6. This bit is writable in Config mode (for Test purposes).
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH1_S*n*FCR) . In other words these bits contain either the start bit (S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appendded to the message is "0". It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM. This bit is writable in Config mode (for Test purposes).

56.3.62 SMC Frame Register n (PSI5_CH1_SFR*n*)

The following figure shows the SMC Frame Register n.

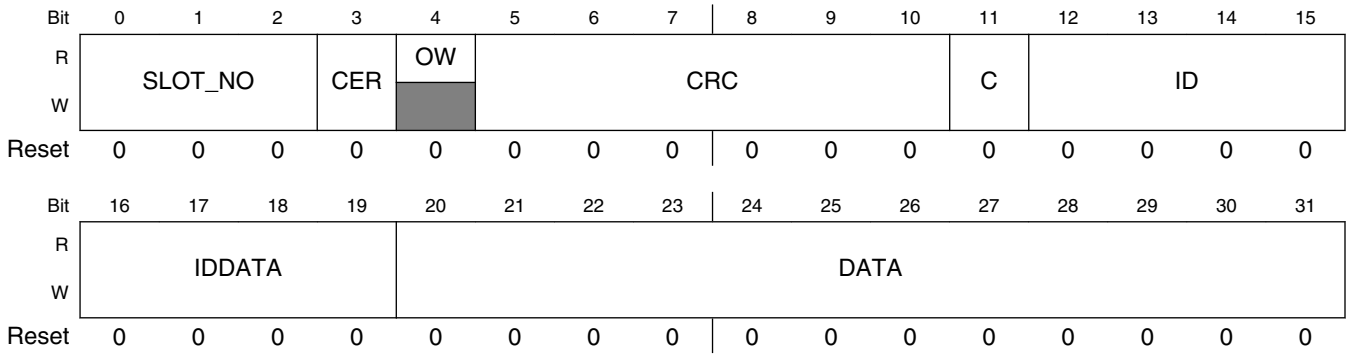
NOTE

Six dedicated 32-bit SFR[x] registers for SMC are available with a 1-1 slot correspondence. These can be read by either the CPU or the DMA (when DMA_EN_SF = 1).

NOTE

No FIFO for SMC frame is available. Six 32-bit registers are available in the address space, which can be accessed by either the DMA (using the DSFR) or the CPU (using the six SFR registers), depending on the setting of the DMA_EN_SMC bit.

Address: 0h base + 2FCh offset + (4d × i), where i=0d to 5d



PSI5_CH1_SFR*n* field descriptions

Field	Description
0–2 SLOT_NO	This indicates in which slot this SMC frame was received. Note that here the slot number refers to the slot in which all the re-arranged PSI5 messages (from which the M0,M1 bits have been extracted) are supposed to have occurred after the correct PSI5 message placement and Timing corrections have been performed. For SFR[1], slot_no = 1; For SFR[2], slot_no = 2; For SFR[3], slot_no = 3; For SFR[4], slot_no = 4; For SFR[5], slot_no = 5; For SFR[6], slot_no = 6; This field is writable in Config mode (for Test purposes).
3 CER	CRC Error This field is writable in Config mode (for Test purposes). 0 The present data has no CRC error. 1 The present data has a CRC error.
4 OW	Overwrite status **NOTE:** If the corresponding SOWSM bits in the SSESR have been set or the OWSM bits in the GISR have been cleared, then the same will be reflected in the message stored in the corresponding RAM location as well. The overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten, the overwrite bit gets set. 0 The present data was already read when new message arrived. 1 New message has overwritten the previous unread message.
5–10 CRC	6-bit CRC for slow serial message

Table continues on the next page...

PSI5_CH1_SFR*n* field descriptions (continued)

Field	Description
	This field is writable in Config mode (for Test purposes).
11 C	Configuration bit This bit is writable in Config mode (for Test purposes). 0 12-bit data and 8-bit message ID. 1 16-bit data and 4-bit message ID.
12–15 ID	Message ID: If C = '0' indicates ID[7:4], if C = '1' indicates ID[3:0]. This field is writable in Config mode (for Test purposes).
16–19 IDDATA	Message ID/DATA: If C = '0' indicates ID[3:0], if C = '1' indicates DATA[15:12]. This field is writable in Config mode (for Test purposes).
20–31 DATA	DATA payload This field is writable in Config mode (for Test purposes).

56.3.63 New Data Status Register (PSI5_CH1_NDSR)

The following figure shows the New Data Status Register.

Address: 0h base + 314h offset = 314h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NDS																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_NDSR field descriptions

Field	Description
0–31 NDS	New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when a new Message arrives i.e. faulty or non faulty in corresponding PSI5 MB location. Bit 0 reflects the MB0 location.These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SNDR.

56.3.64 Overwrite Status Register (PSI5_CH1_OWSR)

Address: 0h base + 318h offset = 318h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OWS																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_OWSR field descriptions

Field	Description
0–31 OWS	Over Write Status flags for PSI5 Messages corresponding to each PSI5 location in the RAM registers. These bits are set when any unread receive PSI5 message is overwritten by a newly arrived message). These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SOWSR.

56.3.65 Error Indication Status Register (PSI5_CH1_EISR)

The following figure shows the Error Indication Status Register.

Address: 0h base + 31Ch offset = 31Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERROR																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_EISR field descriptions

Field	Description
0–31 ERROR	Error Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when any of the five selectable (by ERROR_SELECT) bits (C, E, EM, T or F) is set for the corresponding PSI5 Message. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SEISR.

56.3.66 Set New Data Status Register (PSI5_CH1_SNDSR)

The following figure shows the Set New Data Status Register.

Address: 0h base + 320h offset = 320h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SNDS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_SNDSR field descriptions

Field	Description
0–31 SNDS	Sets New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. The corresponding NDS Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** This bit is writable in config and Normal modes.

56.3.67 Set Overwrite Status Register (PSI5_CH1_SOWSR)

The following figure shows the Set Overwrite Status Register.

Address: 0h base + 324h offset = 324h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SOWS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

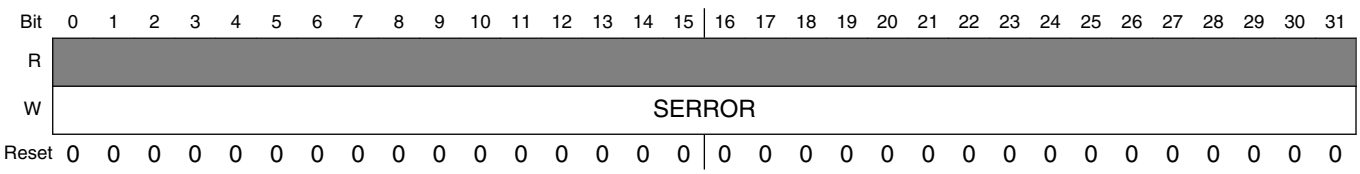
PSI5_CH1_SOWSR field descriptions

Field	Description
0–31 SOWS	Sets overwrite status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding OWS flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **NOTE:** These bits are writable in Config and Normal modes.

56.3.68 Set Error Status Register (PSI5_CH1_SEISR)

The following figure shows the Set Error Status Register.

Address: 0h base + 328h offset = 328h



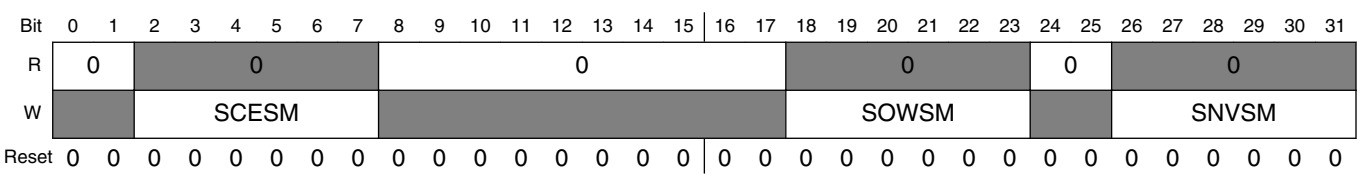
PSI5_CH1_SEISR field descriptions

Field	Description
0–31 SERROR	Sets Error Status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding ERROR flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **NOTE:** These bits are writable in Config and Normal modes.

56.3.69 Set SMC Error Status Register (PSI5_CH1_SSESR)

The following figure shows the Set SMC Error Status Register.

Address: 0h base + 32Ch offset = 32Ch



PSI5_CH1_SSESR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 SCESM	Sets IS_CESM Status flags for SMC Messages corresponding to each SMC MB locations.

Table continues on the next page...

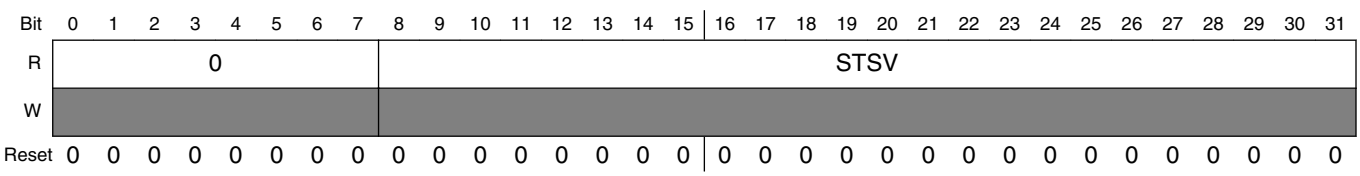
PSI5_CH1_SSESR field descriptions (continued)

Field	Description
	The corresponding IS_CESM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** These bits are writable in Config and Normal modes.
8–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 SOWSM	Sets IS_OWSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_OWSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** These bits are writable in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SNVSM	Sets IS_NVSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_NVSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** This bit is writable in Config and Normal modes.

56.3.70 Sync Time Stamp Read Register (PSI5_CH1_STSRR)

The following figure shows the Sync Time Stamp Read Register.

Address: 0h base + 330h offset = 330h



PSI5_CH1_STSRR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 STSV	Sync Time Stamp Value[23:0] This bitfield provides the value of the free-running 24-bit time stamp counter triggered (captured) at last sync pulse.

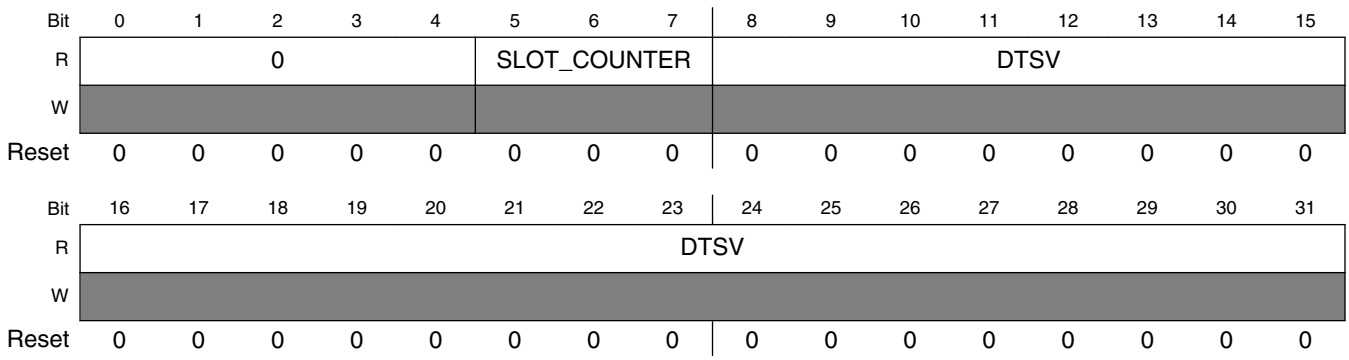
56.3.71 Data Time Stamp Read Register (PSI5_CH1_DTSRR)

The following figure shows the Data Time Stamp Read Register.

NOTE

If S0 and S1 are detected in different slots then the Time Stamp value and the Slot Counter , both will correspond to the first slot(S0)

Address: 0h base + 334h offset = 334h



PSI5_CH1_DTSRR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SLOT_ COUNTER	3bit Slot Counter which corresponds to the slot for which the DTSRR contains the Time Stamp. The slot_counter contains the slot in which the start bits of this frame was captured (after successful detection of S1). If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. In case of asynchronous mode the slot counter has a fixed value of "1".
8–31 DTSV	Data Time Stamp Value[23:0] Whenever a valid sequence of the start bits (S0,S1) is detected then the value of the Time Stamp counter captured at the rising edge of the S0, is stored in this register at the rising edge of S1.

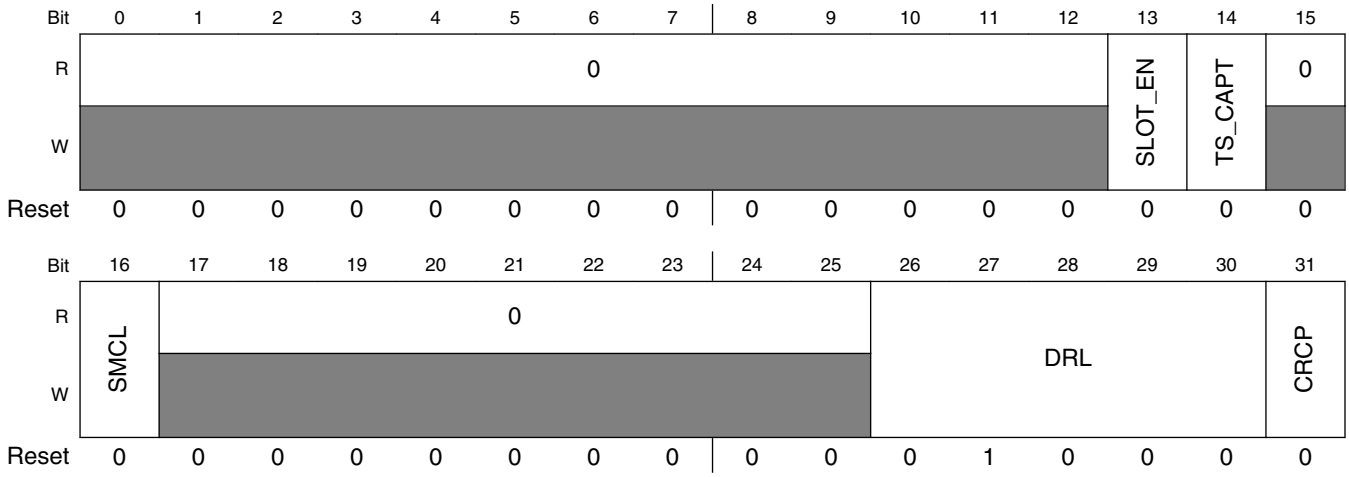
56.3.72 Slot n Frame Configuration Register (PSI5_CH1_S*n*FCR)

The following figure shows the Slot n Frame Configuration Register.

NOTE

In Asynchronous mode, the PSI5_S1FCR register has to be programmed for controlling the various parameters pertaining to the message received. The other registers PSI5_S[2-6]FCR have no effect in the asynchronous mode.

Address: 0h base + 338h offset + (4d × i), where i=0d to 5d



PSI5_CH1_S*n*FCR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SLOT_EN	SLOT_EN 0 This particular slot is disabled i.e not in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) have no effect. 1 This particular slot is enabled i.e is in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) are applicable.
14 TS_CAPT	TS_CAPT 0 Time stamp value captured at rising edge of S0 and stored Start sequence first edge. 1 Time stamp value captured at corresponding Tsync pulse (posedge) and stored with corresponding slot PSI5 message.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SMCL	Serial Messaging Channel field This field is used to enable the detection of start sequence of SMC frame on particular slot. Detection is not done if this field is '0'. 0 Serial Messaging Channel (optional) not present on Rx Message during Time Slot 'n' 1 2-bit{M1,M0} Serial Messaging Channel present on Rx Message during Time Slot 'n'
17–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–30 DRL	Data Region Length[4:0] This field represents Length of Data region (including Frame Control, Status , Data payload1&2 regions and optional messaging bits M0,M1) in Rx Message during Time Slot 'n' 00000-00111: Invalid 01000: 8 bit long data region 01001: 9 bit long data region 01010: 10 bit long data region ....

Table continues on the next page...

PSI5_CH1_S*n*FCR field descriptions (continued)

Field	Description
	11011: 27 bit long data region 11100: 28 bit long data region 11101-11111: Invalid
31 CRCP	CRCP 0 3 bit CRC[2:0] present on Rx Message during Time Slot 'n' 1 1 bit Parity field present on Rx Message during Time Slot 'n'

56.3.73 Slot 1 Start Boundary Register (PSI5_CH1_S1SBR)

The following figure shows the Slot 1 Start Boundary Register.

Address: 0h base + 350h offset = 350h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S1SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_S1SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S1SBT	Slot 1 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 1 should start.

56.3.74 Slot 2 Start Boundary Register (PSI5_CH1_S2SBR)

The following figure shows the Slot 2 Start Boundary Register.

Address: 0h base + 352h offset = 352h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S2SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_S2SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S2SBT	Slot 2 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 2 should start.

56.3.75 Slot 3 Start Boundary Register (PSI5_CH1_S3SBR)

The following figure shows the Slot 3 Start Boundary Register.

Address: 0h base + 354h offset = 354h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S3SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_S3SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S3SBT	Slot 3 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 3 should start.

56.3.76 Slot 4 Start Boundary Register (PSI5_CH1_S4SBR)

The following figure shows the Slot 4 Start Boundary Register.

Address: 0h base + 356h offset = 356h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S4SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_S4SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S4SBT	Slot 4 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 4 should start.

56.3.77 Slot 5 Start Boundary Register (PSI5_CH1_S5SBR)

The following figure shows the Slot 5 Start Boundary Register.

Address: 0h base + 358h offset = 358h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S5SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_S5SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S5SBT	Slot 5 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 5 should start.

56.3.78 Slot 6 Start Boundary Register (PSI5_CH1_S6SBR)

The following figure shows the Slot 6 Start Boundary Register.

Address: 0h base + 35Ah offset = 35Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S6SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_S6SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S6SBT	Slot 6 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 6 should start.

56.3.79 Slot n End Boundary Register (PSI5_CH1_SnEBR)

The following figure shows the Slot *n* Start Boundary Register (PSI5_S *n* SBR).

Slot n End Boundary register is a single register which can be programmed to define the "end" of the nth slot. The n can be defined in the SLOT_NO field of this register.

There are seven boundary registers for the six slots. Six of these registers (S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR) are used to define the start of each of the six slots. The seventh register SnEBR (Slot n End Boundary register) defines the end of the nth slot. The n is programmed in the slot_no field of this register.

The number of slots would automatically be available from the SnEBR. The SnEBR contains the slot number of the last slot and the end boundary of the same. Each slot can separately be enabled or disabled by using the slot_en bit in the SnFCR. When the start slot boundaries are properly defined in the SnSBR (Slot n Start Boundary Register) register but the slot_en bit = 0 in the corresponding SnFCR (Slot n Frame Configuration register) register, then the corresponding slot is treated as "available" but "unconfigured." For such a slot the data/diagnostic bits all remain 0. The "T" bit will, however, get set if a message that was supposed to come in a configured slot actually arrives in an unconfigured slot. Note that the various parameters like the Data length/CRC configuration and so on are valid ONLY for the configured slots. Even when sampling a message in an unconfigured slot, these parameters would be taken from the previous/next configured slot only. It is assumed that no data would be sent in an unconfigured slot. If a data is received in this unconfigured slot it would never be treated as belonging to the unconfigured slot. As a special case if "S0" and "S1" are detected in different slots then a T-bit error is set .

NOTE

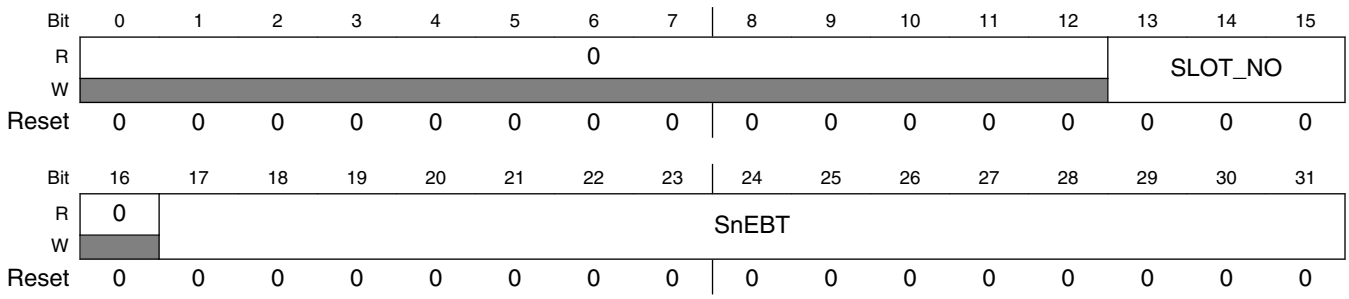
The start slot boundaries in the S *n* SBR should be correctly programmed. If this is not done, the correct operation cannot be guaranteed. Slots occurring after the slot_no value programmed in the SnEBR are treated as being unavailable (that is, the data

in all the slots occurring after the SnEBR is rejected by the Manchester decoder).

Following are the examples:

- If only three slots are required between two sync pulses, program S1SBR, S2SBR, and S3SBR with the correct value of the start boundaries of the three slots and program slot_no = 3 in the S *n* EBR and define the end boundary of the third slot in S *n* EBR. The design ignores slots 4, 5, and 6.
- For six slots, the required programming would be: program the start of each of the six slots in S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Then program the end of the sixth slot in the S *n* EBR by putting slot_no = 6 and define the end boundary of the sixth slot in S *n* EBR.

Address: 0h base + 35Ch offset = 35Ch



PSI5_CH1_SnEBR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 SLOT_NO	The field indicates the Slot number{1-6} for which Slot End Boundary Time is defined. Any value other than 1-6 is defaulted to 0. A value 0 here would mean that the manchester decoder would not sample any slots. This slot number also defines the number of slots that the IP assumes to exist between 2 sync pulses.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 SnEBT	Slot *n* End Boundary Time. This field specifies the time in intervals of 1µs, occurring after the rising edge of timing sync pulse, at which the slot 'n" should end.

56.3.80 Data Output Block Configuration Register (PSI5_CH1_DOBCR)

The following figure shows the Data Output Block Configuration Register.

Table 56-9. CMD_TYPE details

CMD_TYPE	PSI5 Frame [1]	Effect on registers	Writeable Register lengths
"000" - 0	Short Frame(V1.3) with 31 "1s" as the start condition	Command can be written to DPR for auto calculation of stuff/start/CRC bits. Can always be overwritten in DSR/DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	6 bits in DPR 15bits[2] in DBR/DSR.
"001" - 1	Short Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"010" - 2	Long Frame(V1.3) with 31 "1s" as the start condition	-do-	16bits in DPR. 29bits[2] in DBR/DSR.
"011" - 3	Long Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"100" - 4	X-Long Frame(V1.3) with 31 "1s" as the start condition	-do-	22bits in DPR. 37bits[2] in DBR/DSR
"101" - 5	X-Long Frame(V1.3) with 5 "0s" as the start condition.	-do-	-do-
"110" - 6	XX-Long (V2.0) -	-do-	24bits in DPR. 43bits[2] in DBR/DSR
"111" - 7	Non Standard Length.	Command can only be written to DSR and DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	Programmable from 1 to 64 bits in DBR/DSR, depending on the value of PSI5_DOBCR[DATA_LENGTH] field.

1. The start condition(31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command.
2. This is also the length of the command that will be shifted out during the ECU to Sensor communication, regardless of how many "actual" bits are written to DBR/DSR.

Table 56-10. DOBCR bit configurations and the related output states

Output path states	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL
State1	0	1	1
State2[1]	0	0	1
State3[1]	1	1	1
State4[1]	1	0	1
State5[1]	1	x	0

1. These states are shown in Valid states for integrated sync pulse generator. In State5, the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the "sdout" pin.

NOTE

Please refer to Figure 56-26 for the details about the above bits. Figure 56-26 is the actual implementation of Figure 56-25 such

that various contentions between the switches (S1, S2, and so on) do not happen.

Address: 0h base + 360h offset = 360h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA_LENGTH[5:0]						DBR_RST	DSR_RST	CMD_TYPE			DEFAULT_SYNC	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL	SW_READY
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_DOBCR field descriptions

Field	Description
0–5 DATA_LENGTH[5:0]	Can take on values from 0 to 63 corresponding to 1bit to 64bit non standard length commands. When PSI5_DOBCR[CMD_TYPE] == 7(Non Standard) , then this field controls the length of command that can be written to the DBR and the DSR registers, else the value in this field is ignored. This when DATA_LENGTH = 0, then the length of the command that can be written to DSR and the DBR is 1, when DATA_LENGTH = 1, then the length of the command that can be written to DSR and the DBR is 2 and so on, until a maximum of 64 bit length command when DATA_LENGTH = 63. Note that the length of the command that is programmed by using this field, is also equal to the length of the command that shifts out of the DSR register. In other words, it is equal to the number of bits that shift out of DSR register once the DSR_RDY goes low . After these many shifts have happened , then the value corresponding to "DEFAULT_SYNC" starts getting shifted out and DSR_RDY goes as "1".
6 DBR_RST	This is to reset and reject current content to Data Buffer Register. When this bit is written as "1" then the contents of the DBR are reset to all "0s" (if DEFAULT_SYNC = = 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DBR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
7 DSR_RST	This is to reset and reject current content to Data Shift Register.When this bit is written as "1" then the contents of the DSR are reset to all "0s" (if DEFAULT_SYNC = = 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DSR would be ready for new data.Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
8–10 CMD_TYPE	These 3 bits indicate the type of command that needs to be transmitted during the ECU to sensor communication. Table 56-4 is a brief description of the same.
11 DEFAULT_SYNC	When this bit is set to "0" then the default value of DSR and DBR registers are all "0s"; when this bit is set to 1 then the default value of DBR and DSR registers are all "1s". This value is used as the value of the default sync pulses shifted at the output path when the DSR_RDY == 1.
12 GTM_TRIG_SEL	GTM event triggered/internal sync pulse generator selection as shift clock for the DSR. 0 Internal sync pulse generator shift triggered 1 GTM event shift triggered
13 SP_PULSE_SEL	Selects the source for the short pulse PWM module: 0 SP module gets the data from the DSR 1 SP module directly gets input from the GTM_event/internal sync pulse generator

Table continues on the next page...

PSI5_CH1_DOBCR field descriptions (continued)

Field	Description
14 OP_SEL	This bit selects as to which would be driving source of the "ipp_do_psi5_sdout" port. 0 The sync pulse generator select as per the Bit3 1 PWM output
15 SW_READY	When this bit is written to 1 the transfer from DBR to DSR automatically happens as soon as DSR_RDY becomes 1. When this bit is kept to 0 then the transfer from DBR to DSR will remain pending . Once the software makes this bit as "1" and DSR_RDY also goes as "1" the transfer to DSR takes place. However, note that when this bit is 0, and the DSR_RDY becomes 1, still the transfer to the DSR will NOT happen. Only when this bit is a 1 , will the transfer happen. This bit can be written in CONFIG and the Normal modes.

56.3.81 Manchestor Decoder Disable Offset (PSI5_CH1_MDDIS_OFF)

The Manchester Decoder Disable Offset register (PSI5_MDDIS_OFF) defines the time in intervals of 1 µs, for which the Manchester decoder remains disabled after the falling edge of the sync pulse. Using this register, this time can be programmed from 0 µs to 128 µs. This works on the 1 µs clock from the common clock generator module.

Address: 0h base + 362h offset = 362h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								MDDIS_OFF							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_MDDIS_OFF field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MDDIS_OFF	These 7 bits can be used to program the time for which the Manchester Decoder remains disabled. This offset is added AFTER the falling edge of the sync pulse. Thus the total time for which the manchester decoder remains disabled = TsyncH + MDDIS_OFF where TsyncH is the high time for the sync pulse . Thus TsyncH = Pulse_Width0 or Pulse_Width1 as the case maybe, depending on whether a "0" is being pulse modulated or a "1". Note that the Manchester Decoder will ALWAYS remain disabled during the high time of the Sync Pulse. Using this register it is possible to disable the Manchester Decoder BEYOND the Sync Pulse High Time as well. Figure 1032 shows the "MDDIS_OFF" definition.

56.3.82 Pulse Width for Data Bit Value 0 (PSI5_CH1_PW0D)

The following figure shows the Pulse Width for Data Bit 0 Register.

Address: 0h base + 364h offset = 364h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								Pulse_Width0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_PW0D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 Pulse_Width0	This defines the width (in µs) of data value '0' to be send from Data Output Register. It is the number of clock cycles(1µs clock) counted upto width "Pulse_Width0", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. When using PSI5 version 1.3 ECU-to-sensor Communication Pulse_Width0 should be configured to 0. Figure 1032 shows the definition of "Pulse_Width0".

56.3.83 Pulse Width for Data Bit Value 1 (PSI5_CH1_PW1D)

The following figure shows the Pulse Width for Data Bit 1 Register.

NOTE

When using the Version 1.3 format for ECU-to-sensor communication, the pulse width is NOT disabled. In this case, PW0D has to be programmed as 0. PW1D has to have the value of the length of the pulse desired when a 1 has to be transferred.

Address: 0h base + 366h offset = 366h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								Pulse_Width1							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_PW1D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH1_PW1D field descriptions (continued)

Field	Description
9–15 Pulse_Width1	This defines the width (in µs) of data value '1' to be send from Data Output Register. It is the number of clock cycles(1µs clock) counted upto width "Pulse_Width1", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. Figure 1032 shows the definition of "Pulse_Width1".

56.3.84 Counter Target Pulse Register (PSI5_CH1_CTPR)

The following figure shows the Counter Target Pulse Register.

NOTE

Please see Internal SYNC Pulse generation and coordination across PSI5 channels for details about the Sync Pulse generation and coordination.

Address: 0h base + 368h offset = 368h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CTPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_CTPR field descriptions

Field	Description
0–15 CTPR	Counter Target Pulse Register. This is the target counter value. Once the CTC (channel target counter) reaches the value of CTPR this CTC is reset. CTC is a 16 bit free running counter that starts/stops depending on the CTC_GED (in GCR) bit or the CTC_ED bit (in PCCR). It gets reset initially when its value reaches CIPR and subsequently when its value reaches CTPR. The fact that which out of CTC_GED or the CTC_ED bit enables/disables it, is dependent on the CTC_GED_SEL bit in the PCCR. When CTC_GED/CTC_ED == 0 then the CTC is reset. The minimum value of CTPR is 6. Values less than 6 will be defaulted to 0 and the CTPR counter will not start. The value in CTPR indicates the Time Period of the internally generated sync pulses. The un modulated (No PWM) internally generated sync pulses have a high time of 4-sp_ts_clk cycles i.e. the duty cycle of the un modulated internally generated sync pulses is (4/|CTPR| *100). The 4 sp_ts_clk cycles is just the nascent value of the Pulse which actually triggers the PWD counters Note that this gives one more possible state, in addition to the states shown in Valid states for integrated sync pulse generator . This additional state allows the output of the internal sync pulse generator to be made directly available at the output, without being pulse modulated. The CTPR describes ONLY the SYNC pulse period and not the sync pulse length which is governed by the PWD registers, except in State5 where in the GTM governs these parameters.

56.3.85 Counter Initialize Pulse Register (PSI5_CH1_CIPR)

The following figure shows the Counter Initialize Pulse Register.

NOTE

Due to internal Flop synchronisation there can be a jitter of +/- 20 ns between the offset times that are programmed in the CIPR of different channels and the ones actually being observed. Similarly there can be a jitter of +/- 20 ns in the programmed CTPR and the measured CTPR.

NOTE

Please see Internal SYNC Pulse generation and coordination across PSI5 channels for details about the Sync Pulse generation and coordination.

Address: 0h base + 36Ah offset = 36Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CIPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_CIPR field descriptions

Field	Description
0–15 CIPR	Counter initialize pulse register. This is the value using which the CTC gets reset, the first time after which it is enabled using CTC_GED (in GCR) or the CTC_ED bit (in PCCR). All subsequent resets are done when the CTC value reaches CTPR. The CIPR is used to set the offset time between the start of sync pulses between two different PSI5 channels, when the CTC of both of these channels are started simultaneously using CTC_GED. The offset between any two PSI5 channels would be |CIPR_channel1 - CIPR_channel2| sp_ts_clk cycles. This if channel 1 has a CIPR = 0 while channel2 has CIPR = 10, then the offset would be 10 sp_ts_clk cycles.

56.3.86 Data Preparation Register Low (PSI5_CH1_DPRL)

The following figure shows the Data Preparation Register Low.

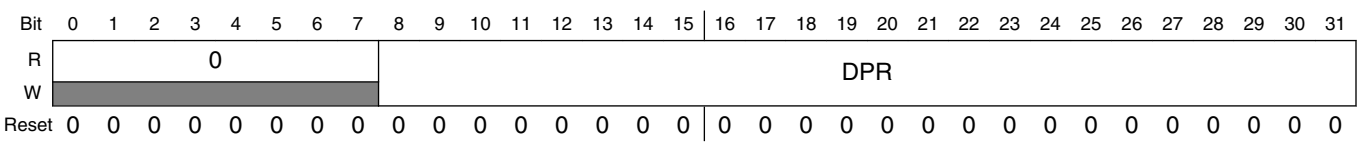
This register is the lower register used for writing the standard data and the address commands and is to be used for writing standard length ECU-to-sensor commands. The CRC/stuff/start bits are automatically appended by the hardware before the command is transferred to DBR. The commands to this register are accepted ONLY when PSI5_GISR[DPR_RDY] == 1. Whenever PSI5_GISR[DPR_RDY] == 1 , then each bit of

this register defaults to 0 . Writing to this register when PSI5_GISR[DPR_RDY] == 0 will cause the PSI5_GISR[IS_PROW] status bit to be set and the generation of an error interrupt, if it is enabled. Further , trying to write to this register when PSI5_DOBCR[CMD_TYPE] = "7" results in the setting of the PSI5_GISR[IS_PROW] bit and the resulting write is rejected.

Table 56-11. DPR details

PSI5_DOBCR[CMD_TYPE]	Standard PSI5 FRAME Types	Writeable bits in PSI5_DPRL
"0" or "1"	Short Frame(PSI5 V1.3)	6(DPR[5:0])
"2" or "3"	Long Frame(PSI5 V1.3)	16(DPR[15:0])
"4" or "5"	X-Long Frame(PSI5 V1.3)	22(DPR[21:0])
"6"	XX-Long Frame(PSI5 V2.0)	24(DPR[23:0])
"7"	Not Available for write	Not Available for write

Address: 0h base + 36Ch offset = 36Ch

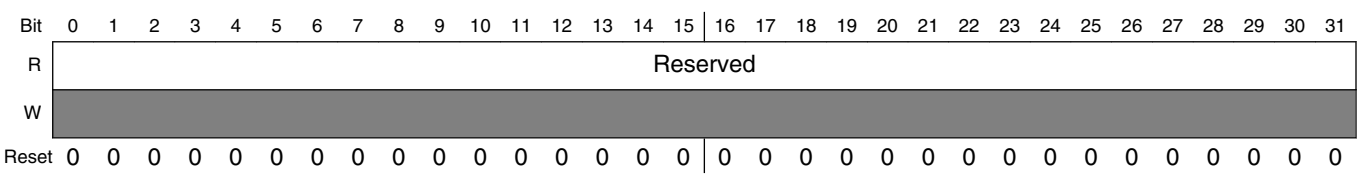


PSI5_CH1_DPRL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 DPR	DPR[23:0] are the 24-bits of the DPR register used for writing the variable length , standard ECU-to-Sensor commands , comprising of the address , data and other fields . Note that the IP is transparent to the arrangement of the address,data and other fields and it treats the bits in these fields identically for the purpose of CRC calculation and transmission. This register is writeable only when using the standard length PSI5 frames . For details about CRC calculation please refer to Data transmission . The writeable bits in this register are governed by the PSI5_DOBCR[CMD_TYPE] bit fields as per Table 56-6 . Note that the unwriteable bits in this register field are always read as "0s" . This register has to be written by the CPU. These bits can be written only in the Normal mode.

56.3.87 Data Preparation Register High (PSI5_CH1_DPRH)

Address: 0h base + 370h offset = 370h



PSI5_CH1_DPRH field descriptions

Field	Description
0–31 Reserved	This field is reserved. This register is always read as 0s. Writing to this register does not have any effect.

56.3.88 Data Buffer Register Low (PSI5_CH1_DBRL)

The following figure shows the Data Buffer Register Low.

This register is writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode.

NOTE

Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1, then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 56-12. DBRL[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" (Non Standard Length)	DBR[31:0]]	Variable length of writeable bits , as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
"6"(XX Long)	DBR[31:30]	DPR[19:18]
	DBR[29]	1'b0(stuff)
	DBR[28:23]	DPR[17:12]
	DBR[22]	1'b0(stuff)
	DBR[21:16	DPR[11:6]
	DBR[15]	1'b0(stuff)
	DBR[14:9]	DPR[5:0
	DBR[8:0]	"0111111110"

Table continues on the next page...

Table 56-12. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"4" or "5" (X-Long)	DBR[31]	1'b1(stuff)
	DBR[30:28]	DPR[20:18]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	DPR[17:15]
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"2" or "3" (Long)	DBR[31:29]	"000"
	DBR[28]	CRC[0]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	{CRC[1],CRC[2],DPR[15]}
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"0" or "1" (Short)	DBR[31:15]	0
	DBR[14:12]	{CRC[0],CRC[1],CRC[2]}[
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]

Table continues on the next page...

Table 56-12. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)

Address: 0h base + 374h offset = 374h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_DBRL field descriptions

Field	Description
0–31 DBR	This register contains the Lower 32 bits, (DBR[31:0]), of the max-64 bit length command (DBR[63:0]). The higher bits, DBR[63:32] are contained in the DBRH register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 56-7 . The table (except when PSI5_DOBCR[CMD_TYPE] = "7") show the bit assignment of the DBR[31:0] when these bits are automatically updated by the hardware. When PSI5_DOBCR[CMD_TYPE] == "7"), then only the CPU can write to these registers.

56.3.89 Data Buffer Register High (PSI5_CH1_DBRH)

The following figure shows the Data Buffer Register High.

This register is CPU writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY]== 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit

PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 56-13. DBRH[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" - Non Standard Frame lengths	DBR[63:32]]	Variable length of writeable bits , as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
"6" - XX Long	DBR[63:43]	All 0s
	DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1], CRC[0]}
	DBR[36]	1'b0(stuff)
	DBR[32:35]	DPR[23:20]
"4" or "5" -X Long	DBR[63:37]	All 0s
	DBR[36]	CRC[0]
	DBR[35]	1'b1((stuff)
	DBR[34:32]	{CRC[1],CRC[2],DPR[21]}
"3" or "2"(Long) OR "1" or "0" (Short)	DBR[63:32]	All 0s

Address: 0h base + 378h offset = 378h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_DBRH field descriptions

Field	Description
0–31 DBR	This register contains the Upper 32 bits(DBR[63:32]) of the max-64 bit length command (DBR[63:0]). The lower bits, DBR[31:0] are contained in the DBRL register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 56-8 . Note that when PSI5_DOBCR[CMD_TYPE] == "7") , then only the CPU can write to these registers.

56.3.90 Data Shift Register Low (PSI5_CH1_DSRL)

The following figure shows the Data Shift Register Low.

This register is CPU writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addtion to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 37Ch offset = 37Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH1_DSRL field descriptions

Field	Description
0–31 DSR	This register contains the lower 32 bits, (DSR[31:0]), of the max-64 bit length command (DSR[63:0]). The higher bits, DSR[63:32], are contained in the DSRH register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits.When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRL . Thus when updated by the hardware, then DSR[31:0] = DBR[31:0] . Hardware updation occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0 then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1 . These bits can be written by the CPU in the Normal mode , when PSI5_GISR[DSR_RDY] == 1'b1 .

56.3.91 Data Shift Register High (PSI5_CH1_DSRH)

The following figure shows the Data Shift Register High.

This register is writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addtion to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 380h offset = 380h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

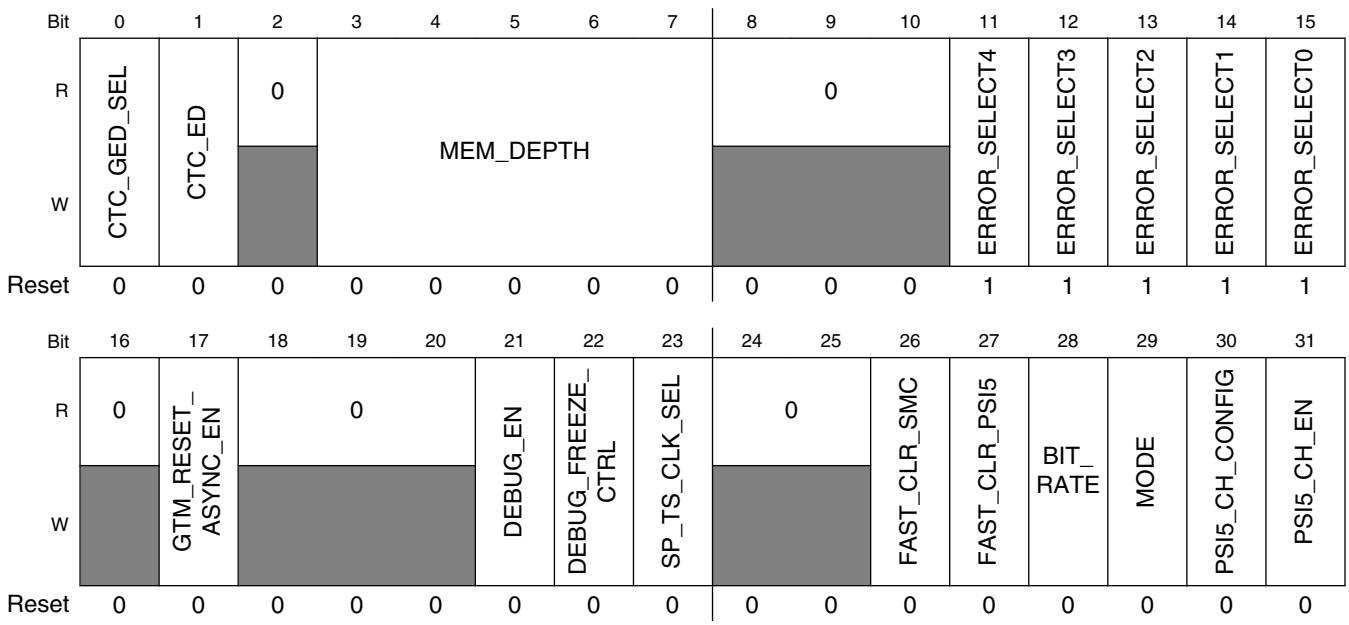
PSI5_CH1_DSRH field descriptions

Field	Description
0–31 DSR	This register contains the higher 32 bits (DSR[63:32) of the max-64 bit length command (DSR[63:0]). The lower bits, DSR[31:0], are contained in the DSRL register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRH . Thus when updated by the hardware, then DSR[63:32]] = DBR[63:32] . Hardware updation occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0 then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1 . These bits can be written by the CPU in the Normal mode , when PSI5_GISR[DSR_RDY] == 1'b1 .

56.3.92 PSI5 Channel Control Register (PSI5_CH2_PCCR)

This section shows the PSI5 Channel Control Register.

Address: 0h base + 388h offset = 388h



PSI5_CH2_PCCR field descriptions

Field	Description
0 CTC_GED_SEL	Channel Target Counter, Global Enable/Disable Select. This bit selects whether the CTC should be enabled/disabled by CTC_GED bit (in GCR) or the CTC_ED bit. Enabling by the CTC_GED bit is used when staggering of the channels is required. 0 CTC enabled disabled by CTC_ED. 1 CTC enabled/disabled by CTC_GED.
1 CTC_ED	Channel Target Counter Enable/Disable. This bit is used to control the CTC locally from the channel in case the CTC_GED_SEL = 0. **NOTE:** This bit is writable in Normal mode and the Config mode. 0 The CTC counters is disabled and reset. 1 The CTC counter is enabled and start counting.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 MEM_DEPTH	Can be programmed from 0 to 31 and denotes the size of the memory that should be used for storing the PSI5 messages. Area above the MEM_DEPTH is treated as being unavailable for message storage. MEM_DEPTH also denotes the number of RAM registers available for the CPU access.

Table continues on the next page...

PSI5_CH2_PCCR field descriptions (continued)

Field	Description
	Note that the same memory can be accessed sequentially (though the DMA) or randomly anytime by the CPU. 0: it indicates that only 1 location of the memory is available for storage 31: it indicates that all the 32 locations of the memory are available for storage. Notes: • Depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR, the MEM_DEPTH memory size may be accessed by CPU only (interrupts are generated) or CPU and DMA (through the dedicated DMA popup registers). • DMA_PM_DS_CONFIG = conf1 then DMA requests are disabled and only the interrupts can be used for data transfer. • DMA_PM_DS_CONFIG = conf2,conf3,conf4 then DMA requests are enabled. The interrupts should be disabled by writing a 0 in the corresponding IE bits. • For more details please refer to description of DMA_PM_DS_CONFIG in the DCRs. • The data in the memory area defined by the MEM_DEPTH parameter is always filled in a sequential fashion as in a FIFO, but can be read sequentially or randomly. There can be no slot-memory location correspondence due to the T bit errors in the manchester decoder. • The memory area above the configured MEM_DEPTH size will always be read as 0. • Irrespective of the configuration, the data can always be randomly read by the CPU from the PMRL/PMRH register.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERROR_ SELECT4	The ERROR_SELECT bitfiield indicates which of the C, E, EM, T, or F error conditions generate an interrupt when the EICR Interrupt Enable bit is set. The individual bits are mapped as {C,E,EM,T,F}. The corresponding error is also latched in the PSI5_EISR. Default: ERROR_SELECT = 11111: Any of C,E,EM,T,or F generate an interrupt. C error CRC Error (C) Bit 20: ERROR_SELECT[4] or C_INT_SEL 0 The C bit does not generate an interrupt. 1 The C bit generates an interrupt.
12 ERROR_ SELECT3	Error_Select[3] or E_INT_SEL E error Electrical Error (E) 0 The E bit does not generate an interrupt. 1 The E bit generates an interrupt.
13 ERROR_ SELECT2	Error_Select[2] or EM_INT_SEL 0 The EM bit does not generate an interrupt. 1 The EM bit generates an interrupt.
14 ERROR_ SELECT1	Error_Select[1] or T_INT_SEL 0 The T bit does not generate an interrupt. 1 The T bit generates an interrupt.

Table continues on the next page...

PSI5_CH2_PCCR field descriptions (continued)

Field	Description
15 ERROR_ SELECT0	Error_Select[0] or F_INT_SEL 0 The F bit does not generate an interrupt. 1 The F bit generates an interrupt.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 GTM_RESET_ ASYNC_EN	GTM_RESET_ASYNC_EN 0 Both the assertion and the deassertion of the GTM reset are treated synchronously. Inside the IP the gtm reset is synchronized on the SP_TS_CLK. 1 Assertion of the GTM reset is treated asynchronously but deassertion is synchronized on the SP_TS_CLK.(For details of SP_TS_CLK, refer to bit 23 of) PSI5 Channel Control Register (PSI5_CH0_PCCR)
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 DEBUG_EN	This bit allows/prevents the IP from entering the debug mode, whenever the debugger is connected and a breakpoint is encountered by the debugger: 0 The IP never enters the debug mode, even if the debugger is connected and a breakpoint is encountered. It keeps on working normally, as in the functional mode. 1 Whenever the debugger is connected and a breakpoint is encountered, the IP enters the debug mode. In the Debug mode, the DMA read pointers and other status registers that normally get affected by the READ operation remain unaffected.
22 DEBUG_ FREEZE_CTRL	DEBUG_FREEZE_CTRL 0 When the IP enters the Debug mode then it goes into the “Open” mode, i.e., the core of the IP keeps on working normally. In this case, there can be a difference between the value of a particular status bit, which is there at the time Debug mode is entered and the time at which the debugger reads it, since some upcoming events might change the status. The hardware clearing, of course, remains disabled. **NOTE:** In the above configuration, the status of registers read through the debugger may be different when entering debug mode and when actually accessing registers at some later point of time. 1 As soon as debug mode is asserted, the state of the system registers that are affected by the hardware gets frozen to what ever it was existing at the point at which the Debug mode goes active.This can be useful for reading the state of the system at a particular point of time such that it is NOT affected by the next upcoming events. Note that Manchester decoder/Sync Pulse Generator/ Time Stamp counters keep on working normally but the registers (data/status/timestamp) affected by these are not updated. When the IP enters the “Debug mode” then it goes into the “Freeze Mode”. Note that whenever there is a request to enter the freeze mode then the IP enters this mode, ONLY when the Manchester decoder and the Sync Pulse Generator are stopped coherently. This means that before entering the “freeze mode”, if there is any pending reception that is ongoing, then the same is completed first. Simultaneously the “sdout” line of the IP should be at a low level. Once these two conditions are satisfied, the IP enters the Debug Freeze mode. The Time Stamp counters are also halted. The fact that the freeze mode has been entered, can be checked by reading the GISR[IS_DB_FR] bit. Once the IP comes out of the freeze mode then this bit goes to 0. **NOTE:** In the above configuration, incoming packets could get lost while the Debug mode is active.
23 SP_TS_CLK_ SEL	This bit controls which clock goes to the Sync Pulse and Time Stamp Generator Unit.

Table continues on the next page...

PSI5_CH2_PCCR field descriptions (continued)

Field	Description
	NOTE: The above selection affects ONLY the clock for the Time Stamp Unit and Sync Pulse Generator. The 1 MHz clock to the rest of the logic like the Manchester decoder or the Slot Boundary Counters is still the 1 MHz from the common clock generator module. **NOTE:** The clock selected using this bit is called SP_TS_CLK. 0 The 1 MHz clock generated from the common clock generator module goes for clocking the Time Stamp Unit and sync pulse generator. 1 The clock generated from the GTM goes for clocking the Time Stamp Unit and Sync Pulse Generator.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FAST_CLR_SMC	This bit controls the clearing mechanism of the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits in the GISR. **NOTE:** This bit is effective only when DCR[DMA_EN_SF] = 1. When FAST_CLR_SMC = 1 then the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits are automatically cleared upon reading the DSFR message register, as per the slot[x] message read in from the DSFR. When reading directly from the SFR registers, this bit does not affect auto clearing. If FAST_CLR_SMC = 0 then the above bits have to be cleared by a w1c. 0 Fast clearing is disabled (clear when written to 1). 1 Fast clearing is enabled.
27 FAST_CLR_PSI5	This bit control the clearing mechanism of bits in NDSR(in conf2, conf3 and conf4 modes), EISR (in conf2, conf3 and conf4 modes), and OWSR(in conf2 and conf3 modes). Following are the details in the various modes of the DCR[DMA_PM_DS_CONFIG] bits: • In conf1 mode: This bit has no effect. Reading the DDSR and the DPMR returns a 0. • In conf2 mode: In this case the diagnostic bits (EISR, NDSR) are cleared ONLY when the diagnostic registers appended with the corresponding PSI5 messages are completely read using the DPMR. During the intermediate operation (when the PSI5 message has been read but the diagnostic bits are still pending), the PSI5 message read from the DPMR will not clear these bits. Both of these registers are simultaneously cleared when the corresponding DMA operation is over. However, only those bits are cleared, the corresponding messages of which form the part of the DMA request being serviced. Note that the OWSR bits are cleared upon the corresponding PSI5 message read from the DPMR. Reading the DDSR returns a 0 in this mode. • In conf3 mode: The PSI5 diagnostic register bits (EISR,NDSR) and OWSR are automatically cleared when the corresponding PSI5 message is read from the DPMR. The specific register bits of EISR,NDSR and OWSR, corresponding to each message, are cleared sequentially one by one as the corresponding message is being read by the DMA. Note that the three bits (each of EISR, NDSR, and OWSR) corresesponding to a specific location are always cleared simultaneoulsy. Reading the DDSR returns a 0. When reading from the PMRH/PMRL registers, this bit does not have any effect on fast clearing. • In conf4 mode: In this case the diagnostic bits (EISR, NDSR) are cleared as soon as these registers are read through the DDSR. These registers are cleared simultaneously in one go, as soon as the corresponding DMA request has been serviced. Only those bits of these registers are cleared, the messages of which form a part of the DMA request. Note that in this mode the reading of the PSI5 messages from the DMA is not available as the request is being used for reading the diagnostic registers. Reading the DPMR returns a 0. The PSI5 messages can be read only from the PMRH/PMRL register. Hence the OWSR has to be cleared only by w1c. • In all the above modes, when reading from the PMRH/PMRL registers, the FAST_CLR_PSI5 does not have any effect on fast clearing. • Further, In all the above cases if FAST_CLR_PSI5 = 0 then the specific diagnostic bits and the OWSR have to be cleared by w1c.

Table continues on the next page...

PSI5_CH2_PCCR field descriptions (continued)

Field	Description
	NOTE: This bit is effective only when DMA_PM_DS_CONFIG = conf2, conf3, or conf4 else this bit has no effect and only software clearing is available. Further, though the OWSR is not directly read by the DMA as a single register, it is cleared in specific modes (when FAST_CLR_PSI5 == 1) as can be seen below. 1 Fast Clearing Enabled 0 Fast Clearing Disables (clear when written to 1)
28 BIT_RATE	This bit selects the receive message bit rate (T bit) for this particular PSI5 Channel, that is, it selects one of the two driven clocks (4 MHz or 6.048 MHz) in the common clock generator module. 0 125 Kbit/s bit rate selected (4 MHz clock). 1 189 Kbit/s bit rate selected (6.048 MHz clock).
29 MODE	This bit selects the operating modes. 0 Asynchronous operating mode (Integrated sync pulse generator module is switched off, only RX active). Note that in this mode the sync pulses from the internal sync pulse generator or the GTM are not allowed to be propagated to the output path, which always remains at logic 0. There is no T-bit error in the Asynchronous mode. 1 Synchronous operating modes (Integrated sync pulse generator module is switched on, both TX and RX subblocks active).
30 PSI5_CH_ CONFIG	PSI5 Channel Config mode request. If this bit is set (and PCCR[PSI5_CH_EN] = '1')this particular psi5 channel enters Configuration mode from Disable mode. Default value is 0. User software must program all PSI5 channel parameters before clearing this bit. Once cleared, channel parameters in registers are locked for writing. The values set can not be changed until this bit is set again after entering the Disable mode **NOTE:** This bit can be written in all modes, but writing a 0 in the Normal mode does not change the operation mode of the device. For more details please see Figure 56-2 and Table 56-2 . Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written. 1 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') - PSI5 Channel enters Config mode. Various configuration registers can be programmed in this mode. The only registers NOT writable in the config mode but writable in the Normal mode are the output registers PSI5_DBR and the PSI5_DSR. 0 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') PSI5 Channel enters Normal mode. Channel parameter registers are locked for writing. This bit can be written in all the modes but writing a "0" in the Normal mode will not change the operation mode of the device.
31 PSI5_CH_EN	PSI5 Channel Enable. If this bit is cleared this particular psi5 channel continues to stay in Disable mode even if GCR[GLOBAL_DISABLE_REQ] = 0. If set, it enters Configuration mode from Disable mode (based on PCCR[PSI5_CH_CONFIG]). Default value is 0. All state machines are disabled. This bit can be written in all modes. **NOTE:** When PSI5_CH_EN == 0 then the clock enable for that particular channel that goes from the IP, goes to "0" (inactive state). This also means that the only clock available to the IP in the config mode is the "ipg_clk_s".The enable for the GTM Clock is not controlled by the IP and should be controlled externally. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written.

Table continues on the next page...

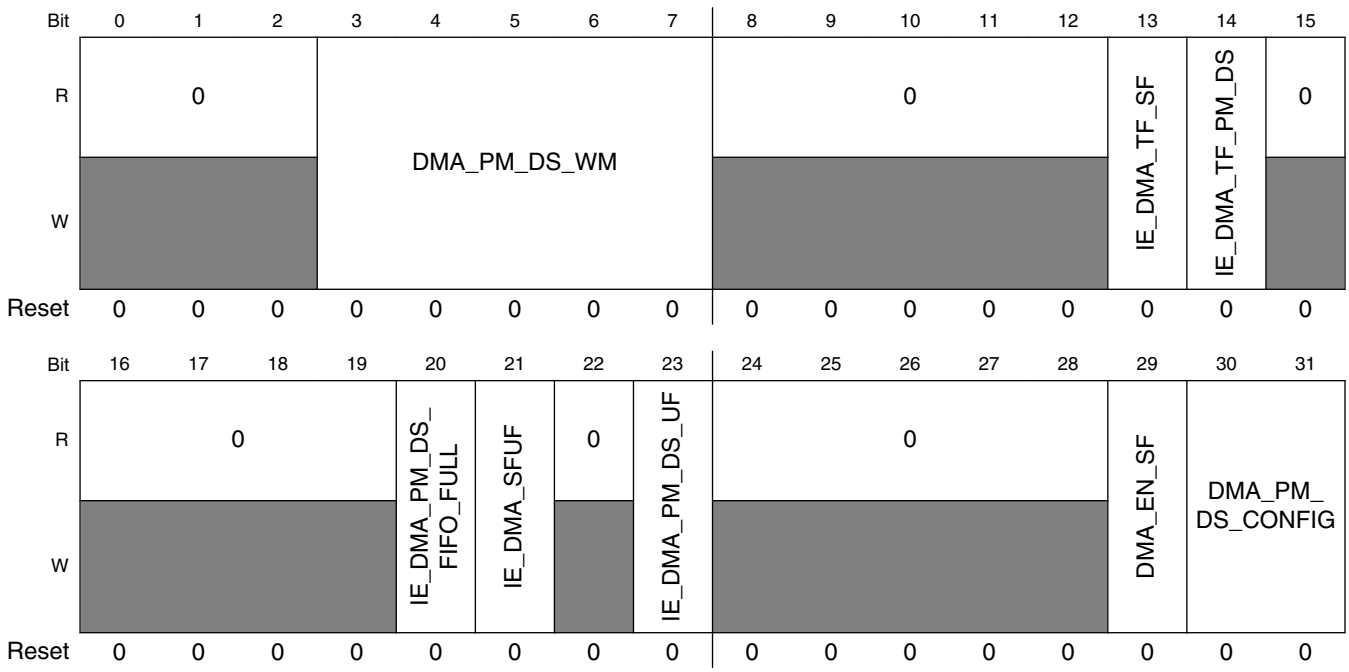
PSI5_CH2_PCCR field descriptions (continued)

Field	Description
	0 PSI5 channel continues in Disable mode. 1 PSI5 channel enabled to enter Config/Normal mode.

56.3.93 DMA Control Register (PSI5_CH2_DCR)

This section defines the DMA Control Register.

Address: 0h base + 38Ch offset = 38Ch



PSI5_CH2_DCR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 DMA_PM_DS_WM	0 to 31: Valid water mark levels. Value fixed by user software. Since each location has two 32-bit registers hence a total of sixty-four 32-bit words are present. The default and minimum value of dma_pm_ds_wm is '0', i.e. 1 message/Diagnostic bit of 1 location. The maximum value of dma_pm_ds_wm is 31, i.e. the actual watermark = 32 RAM locations. **NOTE:** • These bits are ineffective if DMA_PM_DS_CONFIG = conf1. • These bits indicates the number of PSI5 message to be stored in FIFO / number of unread diagnostic bits in the diagnostic registers with a correspondence to the messages in the FIFO,

Table continues on the next page...

PSI5_CH2_DCR field descriptions (continued)

Field	Description
	before the corresponding DMA request is asserted. The watermark definition changes depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR as mentioned below. • When DMA_PM_DS_CONFIG = conf2 or conf3 then the watermark refers to the number of new unread PSI5 messages that are stored in the FIFO before a DMA request is asserted. • In conf2 mode of DMA_PM_DS_CONFIG, the number of words to be transferred, set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value + 2 words (for the two 32-bit diagnostic registers). • In conf3, the number of words to be transferred set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value. • When DMA_PM_DS_CONFIG = conf4 then the watermark refers to the number of new unread diagnostic bits in the diagnostic register. In this configuration the PSI5 messages can only be read by the CPU, since the DMA request would cater to reading the diagnostic bits only. • In conf4, the number of words transferred in each DMA request = 2 x 32-bit words (for the two 32-bit diagnostic registers). The water mark however refers to the number of new unread diagnostic bits pertaining to each PSI5 message in the FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IE_DMA_TF_SF	**NOTE:** This bit is writeable in Config and Normal modes. Enable for interrupt, which is generated when DMA transfer finishes for DMA SMC Frame register. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_DMA_TF_ PM_DS	Enable for Interrupt, which is generated when DMA transfer finishes for PSI5 messages/DMA Diagnostic Status register depending on the DMA_PM_DS_CONFIG bits of the DCR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IE_DMA_PM_ DS_FIFO_FULL	This bit is effective only when the DMA_PM_DS_CONFIG = conf2,conf3 or conf4. It enables the FIFO FULL condition generation interrupt. For the cases which generate these FIFO FULL conditions please refer to the description of IS_DMA_PM_DS_FIFO_FULL register bit of the DSR. **NOTE:** This bit is writable in Config and Normal modes.
21 IE_DMA_SFUF	Enables interrupt when there is underflow in DMA SMC Frame register when DMA is enabled. It is set when the DSFR is read without a valid DMA request, i.e. it is empty. Default value is 0. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH2_DCR field descriptions (continued)

Field	Description
23 IE_DMA_PM_DS_UF	Enables interrupt when there is underflow in DMA PSI5 message register when DMA is enabled. Default value is 0. For the details as to when this underflow occurs, please refer to the details of IS_DMA_PM_DS_UF bit details of the DSR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DMA_EN_SF	Enable DMA request for SMC Frame Data 0 DMA for SMC frame is disabled. The six dedicated 1-1 slot corresponding SMC registers can only be read by the CPU through the SFR. Reading the DSFRs returns 0. 1 DMA for SMC frame is enabled. The six dedicated slot corresponding SMC registers can be read by reading the DSFR successively. In the DMA mode the six registers are read in a round robin fashion as explained in the DSFR details. Reading directly through the SFR registers is still available.
30–31 DMA_PM_DS_CONFIG	These bits define how the PSI5 messages are stored in the MEM_DEPTH memory area and the associated diagnostic bits (EISR, NDSR) are transferred. The DPMR, DDSR, and DSFR registers should be used for DMA access and the PMRH/PMRL, NDSR, EISR, and OWSR registers should be used for CPU access. **NOTE:** For more details please see the description of the DCR[DMA_PM_DS_WM] bits. When DMA_PM_DS_CONFIG bits = conf2 or conf3 and if the CPU and the DMA are simultaneously used for reading the PSI5 messages, then in the course of reading the messages it is possible that there can be an overflow, though some intermediate locations, which have been read by the CPU, are empty. This will occur because the DMA access is made independent of the CPU access. The DMA logic would never know that the CPU has already read a message which was scheduled to be read by the DMA. Whenever the DMA is used for reading the PSI5 message (conf2/conf3) modes then though the CPU can read the messages in parallel through the PMRH/PMRL registers, still for the purpose of setting the Overwrite/Overflow bits only a read from the DMA through the popup registers (DPMR) would be treated as valid read. This will also be true for the SMC messages when accessed simultaneously by the CPU/DMA. **NOTE:** These bits are writable only in config mode. 00 **conf1** The DMA request is disabled. In this mode, only the interrupts can be used for indicating that data transfer is required. 01 **conf2** After transferring the "dma_pm_ds_wm" number of PSI5 messages, there will be two additional 32-bit transfers in which the NDSR and the EISR registers are sequentially transferred through the DPMR registers by using DMA. 10 **conf3** "dma_pm_ds_wm" number of PSI5 messages is transferred in each DMA request. The DMA read is through the DPMR registers. 11 **conf4** Only the diagnostic bits are transferred through the DDSR. The DMA read is through the DDSR registers.

56.3.94 DMA Status Register (PSI5_CH2_DSR)

This section defines the DMA Status Register.

Address: 0h base + 390h offset = 390h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													IS_DMA_TF_SF	IS_DMA_TF_PM_DS	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				IS_DMA_PM_DS_FIFO_FULL	IS_DMA_SFUF	0	IS_DMA_PM_DS_UF	0							
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DSR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IS_DMA_TF_SF	This flag is set when DMA transfer finishes for DMA SMC Frame register.
14 IS_DMA_TF_PM_DS	This flag is set when DMA transfer finishes . For the various configurations of this request please refer to the descriptions of the DMA_PM_DS_CONFIG bits in the DCR. This flag is cleared by w1c.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IS_DMA_PM_DS_FIFO_FULL	Interrupt Status when there is FIFO FULL corresponding to DMA request. The FIFO FULL scenarios are detailed below with respect to the various settings of the DMA_PM_DS_CONFIG bit. **conf1:** FIFO FULL disabled **conf2 and conf3:** FIFO FULL occurs when all the memory locations become full in the PSI5 message FIFO. It indicates that without any read, any new upcoming message will now overwrite the existing messages and set the overwrite bits. This bit is cleared either by w1c or upon a PSI5 message read depending on the FAST_CLR_PSI5 bit. **conf4:** FIFO FULL occurs when there is a "REGISTER FULL" in the Diagnostic registers. FIFO FULL here means that the diagnostic registers contain 32 unread diagnostic bits. A new message starts changing the status of the diagnostic bits. In this configuration it is necessary to read the diagnostic bits through the DMA using the DDSR in order to prevent FIFO FULL. Of course the diagnostic registers NDSR/EISR can be read any time by the CPU, but to prevent FIFO FULL they HAVE to be read through the DDSR registers only. Also in this mode the PSI5 messages DONT contribute to the FIFO FULL. The PSI5 messages have to be read by the CPU using the PMRH/PMRL registers in order to prevent the overwrite bits from being set in the OWSR. In this mode the OWSR bits are NOT automatically cleared upon the PSI5 message read since the PSI5 message has to be read by the PMRH/PMRL register and NOT though the DPMR. The OWSR bits have to be cleared by a w1c. Reading the DPMR returns a 0 in this mode. This bit is cleared by a w1c. 0 No FIFO full. 1 FIFO full has occurred
21 IS_DMA_SFUF	SMC Frame DMA underflow: This happens when the DSFR has been read without a proper DMA request being asserted. The DSFR is empty and it is read. This bit is cleared by a w1c. 0 No underflow has occurred. 1 Underflow has occurred.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IS_DMA_PM_DS_UF	Depending on the DMA_PM_DS_CONFIG bits following is the underflow conditions: Default value is 0. **conf1:** underflow disabled **conf2 and conf3:** Underflow happens when the software reads the PSI5 message FIFO though the DPMR, beyond the available messages (empty area). **conf4:** Underflow happens when the diagnostic registers are read through the DDSR without a valid DMA request and the DDSR contains no new data. The bits are cleared by a w1c.

Table continues on the next page...

PSI5_CH2_DSR field descriptions (continued)

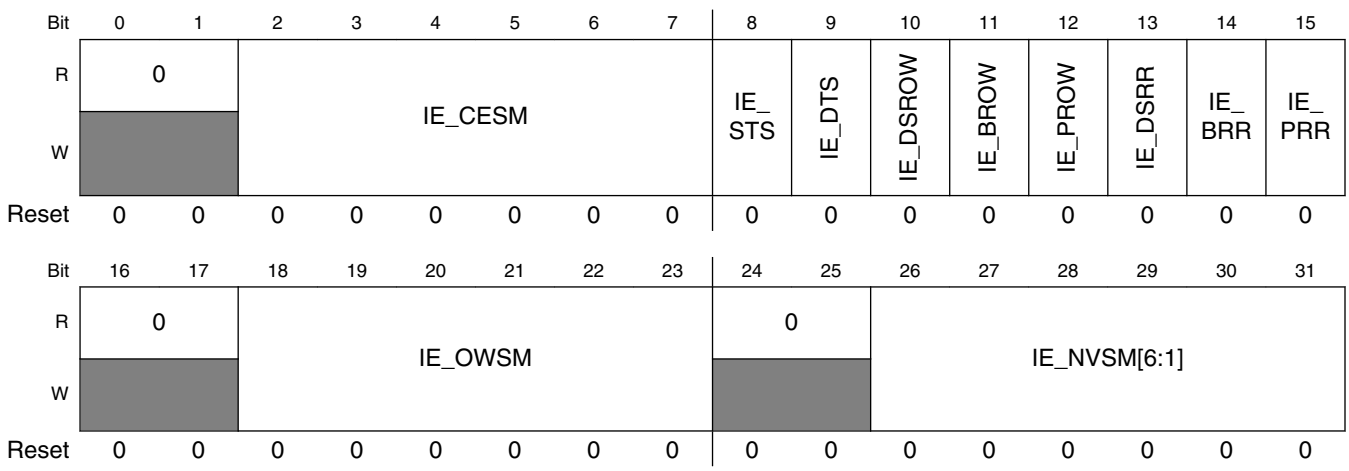
Field	Description
	0 No underflow has occurred. 1 Underflow has occurred.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.95 General Interrupt Control Register (PSI5_CH2_GICR)

The GICR contains the bits for controlling interrupts related to:

- SMC messages
- ECU-to-sensor communication
- the Time Stamp

Address: 0h base + 394h offset = 394h



PSI5_CH2_GICR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IE_CESM	Interrupt request when received SMC frame in corresponding slot has CRC failure (CRC recalculation on SMC). **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 IE_STS	Interrupt request enabled when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR. **NOTE:** This bit is writable in Config and Normal modes.

Table continues on the next page...

PSI5_CH2_GICR field descriptions (continued)

Field	Description
	0 Interrupt is disabled. 1 Interrupt is enabled.
9 IE_DTS	Interrupt request enabled when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_DSROW	Interrupt request enabled when system tries to overwrite on Data Shift register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 IE_BROW	Interrupt request enabled when system tries to overwrite on buffer register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_PROW	Interrupt request enabled when system tries to overwrite on Preparation register when it is not ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_DSRR	Interrupt request enabled when Data Shift Register is ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_BRR	Interrupt request enabled when buffer Register is ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 IE_PRR	Interrupt request enabled when Preparation Register ready to accept new data. **NOTE:** This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IE_OWSM	Interrupt request enable for the SMC message Overwrite bits.

Table continues on the next page...

PSI5_CH2_GICR field descriptions (continued)

Field	Description
	The interrupt for SMC message overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten the overwrite bit gets set. **NOTE:** These bits are writable in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IE_NVSM[6:1]	Interrupt request enabled when any new valid SMC message (fault free) is received in SFR[i]{i:0 to 5} i.e. during slot1 to slot6. This interrupt is generated only when corresponding IS_NVSMS[5:0] is set. **NOTE:** These bits are writable in Config and Normal modes. For each bit position, the corresponding request is enabled as follows: 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.96 New Data Interrupt Control Register (PSI5_CH2_NDICR)

The NDICR contains the interrupt enable bits related to a new PSI5 message arrival, in the corresponding PSI5 message buffer location.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 398h offset = 398h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ND																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_NDICR field descriptions

Field	Description
0–31 IE_ND	Interrupt request enabled when any new message (fault-free/with fault) is received in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.97 Overwrite Interrupt Control Register (PSI5_CH2_OWICR)

The Overwrite Interrupt Control Register (PSI5_OWICR) is related to the generation of the overwrite interrupts. The overwrite interrupt is generated when the unread PSI5 message in a particular RAM location is overwritten by another PSI5 message.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 39Ch offset = 39Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_OW																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_OWICR field descriptions

Field	Description
0–31 IE_OW	Interrupt request enabled when any new message overwrites the old unread PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt generated only when corresponding RAM buffer Ready with new data. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.98 Error Interrupt Control Register (PSI5_CH2_EICR)

The following figure shows the Error Interrupt Control Register.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 3A0h offset = 3A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ERROR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_EICR field descriptions

Field	Description
0–31 IE_ERROR	Interrupt request enabled when any/all of the error conditions C, E, EM, T, or F is observed in a PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when

PSI5_CH2_EICR field descriptions (continued)

Field	Description
	corresponding RAM buffer Ready with new data. Which of the error condition (any of C, E, EM, T, or F) would generate the interrupt is selectable by PCCR[ERROR_SELECT] field. If all are selected then which out of C,E,EM,T,F is set can be isolated by reading corresponding received message. **NOTE:** These bits are writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

56.3.99 General Interrupt Status Register (PSI5_CH2_GISR)

The GISR contains the status bits related to SMC messages, ECU-to-sensor communication, and the Time Stamp.

Address: 0h base + 3A4h offset = 3A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IS_DB_FR	0	IS_CESM						IS_STS	IS_DTS	IS_DSROW	IS_BROW	IS_PROW	DSR_RDY	DBR_RDY	DPR_RDY
W			w1c						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		IS_OWSM						0		IS_NVSM					
W			w1c								w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_GISR field descriptions

Field	Description
0 IS_DB_FR	This flag is set to “1” when the IP enters the Debug freeze mode. Please see Debug mode for details about the Debug mode. This bit is auto cleared by the hardware when the IP exits the debug freeze mode. 0 IP not in debug freeze mode. 1 IP in debug freeze mode.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IS_CESM	These flags are set when the received SMC frame in the corresponding slot has a CRC failure (CRC recalculation on SMC). These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.
8 IS_STS	This Interrupt flag is set when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR.
9 IS_DTS	This Interrupt flag is set when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR.
10 IS_DSROW	This flag is set when the system tries to overwrite on Data Shift register when it is not ready to accept new data.
11 IS_BROW	This flag is set when the system tries to overwrite on buffer register when it is not ready to accept new data.
12 IS_PROW	This flag is set when the system tries to overwrite on Preparation register when it is not ready to accept new data.
13 DSR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Shift register is ready to receive new data from the system bus side or the Data Buffer register. When "0" it indicates that there is a ongoing command transmission of the data in DSR , or the CPU has done a "w1c" to this bit leading to the new command transmission . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_DSROW] bit gets set. A write to PSI5_DOBCR[DSR_RST] will abort the current command transmission and make the DSR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, and PSI5_DOBCR[SW_READY] == 1 , then it indicates to the IP that the data in DSR is valid and is to be used for command shifting . This procedure is used when the data is written by the CPU to DSR directly skipping the DBR. For detailed action of this bit please refer to Data transmission . **NOTE:** For this field, Individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. **NOTE:** In configuration mode, writes to this bit are ignored.
14 DBR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Buffer register is ready to receive new data from the system bus side or the Data Preparation register. When "0" it indicates that there is a ongoing shift of the data from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_BROW] bit gets set. A write to PSI5_DOBCR[DBR_RST] will abort the current data transaction in DBR and make the DBR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DBR is valid and is to be used for shifting to DSR . This procedure is used when the data is written by the CPU to DBR directly skipping the DPR. For detailed action of this bit please refer to Data transmission . **NOTE:** For this field, individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used.

Table continues on the next page...

PSI5_CH2_GISR field descriptions (continued)

Field	Description
	NOTE: In configuration mode, writes to this bit are ignored.
15 DPR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Preparation register is ready to receive new data from the system bus side . When "0" it indicates that there is a ongoing processing of the data in DPR or there is a data shift ongoing from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_PROW] bit gets set. When PSI5_DOBCR[CMD_TYPE] == "7" then this bit remains at value "0". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DPR is valid and is to be used for processing and shifting to DBR . For detailed action of this bit please refer to Data transmission . **NOTE:** For these register bits individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. **NOTE:** In configuration mode, writes to this bit are ignored.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IS_OWSM	Status for the SMC message overwrite bits. Each of the 6 locations correspond to the SMC message pertaining to each SFR[x] register, x = 1 to 6. These bits are automatically cleared upon message read when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IS_NVSM	These flags are set when corresponding SFR[i] receives and is ready (to read) with new valid SMC message. These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1.

56.3.100 DMA PSI5 Message Register (PSI5_CH2_DPMR)

This section defines the DMA PSI5 Message Register.

Address: 0h base + 3A8h offset = 3A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DPMR field descriptions

Field	Description
0–31 PSI5_RXDATA	PSI5_RXDATA[31:0]

PSI5_CH2_DPMR field descriptions (continued)

Field	Description
	This popup register contain the PMRL[31:0] data followed by the PMRH[31:0] data for each of the FIFO locations, when the FIFO is read by the DMA. It would require two 32-bit IPS access to read one PSI5 Message. For each of the locations the DATA is pushed sequentially till all the data corresponding to the programmed watermark is complete after which the DMA request goes down. The depth of the FIFO is 32x64. This popup register can be used differently based on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. When DMA_PM_DS_CONFIG=conf2 then it pops the PSI5 message followed by NDSR followed by EISR. When DMA_PM_DS_CONFIG= conf3, it pops only the PSI5 messages. When DMA_PM_DS_CONFIG=conf4 then it pops the NDSR followed by EISR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH2_PCCR) , DMA Control Register (PSI5_CH2_DCR) and DMA Status Register (PSI5_CH2_DSR) .

56.3.101 DMA SMC Frame Register (PSI5_CH2_DSFR)

This section shows the DMA SMC Frame Register.

Address: 0h base + 3ACh offset = 3ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SMC_RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DSFR field descriptions

Field	Description
0–31 SMC_RXDATA	When the DMA_EN_SF = 1 then the six SFR registers are searched in a round robin fashion for the reception of the complete SMC data. The DMA request is asserted as soon as the first encountered SFR has a complete SMC frame. This request remains asserted till all the registers which have a complete data ready have transferred the data through the DMA. Each other SFR is a 32-bit register. This register is a reflection of the location, pointed to by the DMA read pointer.

56.3.102 DMA Diagnostic Status Register (PSI5_CH2_DDSR)

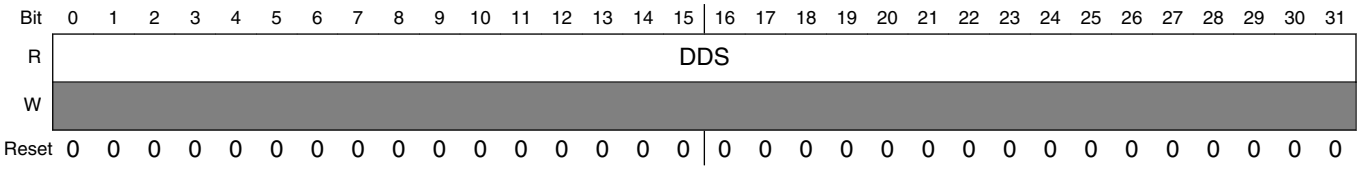
This section defines the DMA Diagnostic Status Register.

NOTE

No FIFO is implemented for diagnostic registers. They are two 32-bit registers addressable by either the CPU (reading NDSR

or EISR directly) or the DMA (reading the DDSR successively).

Address: 0h base + 3B0h offset = 3B0h



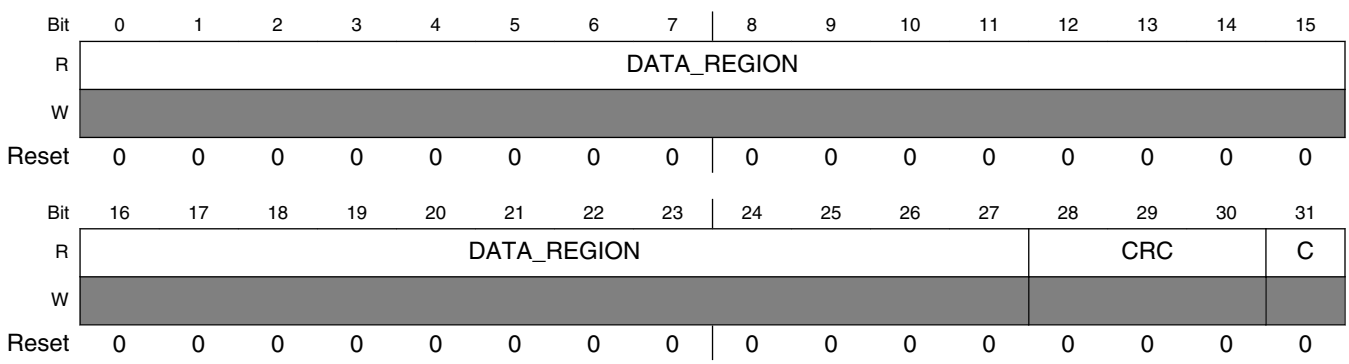
PSI5_CH2_DDSR field descriptions

Field	Description
0–31 DDS	This register maps each of the individual Diagnostic registers (in the order NDSR, EISR) depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH2_PCCR) , DMA Control Register (PSI5_CH2_DCR) and DMA Status Register (PSI5_CH2_DSR) .

56.3.103 PSI5 Message Receive Register Low (PSI5_CH2_PMRRL)

This section defines the PSI5 Message Receive Register Low.

Address: 0h base + 3B4h offset = 3B4h



PSI5_CH2_PMRRL field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field.

Table continues on the next page...

PSI5_CH2_PMRRL field descriptions (continued)

Field	Description
28–30 CRC	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

56.3.104 PSI5 Message Receive Register High (PSI5_CH2_PMRRH)

This section defines the PSI5 Message Receive Register High.

NOTE

Any new PSI5 message is always stored in the PSI5 receive register (PMRRL/PMRRH). In parallel it is also stored in the RAM registers in a sequential manner like in a Ring Buffer. When a new message comes it would always overwrite the data in the PSI5 receive register. But the original message would still be available in the RAM register provided all locations in the RAM registers are not full, and an overwrite has not occured.

Address: 0h base + 3B8h offset = 3B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F	EM	E	T	SlotCounter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_PMRRH field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 F	This represents that NO frame was received in the corresponding configured slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table continues on the next page...

PSI5_CH2_PMRRH field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0,M1), that is, at least one bit is absent during the T bit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This bit indicates electrical error, i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1, and start fields. 1 Electrical error in fields other than M0, M1, and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot, spread across two slots, started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 SlotCounter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,....Slot6.
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH2_S*n*FCR) . In other words these bits contain either the start bit(S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appendded to the message is "0" . It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM.

56.3.105 PSI5 Message Register Low i (PSI5_CH2_PMRL*n*)

The following figure shows the PSI5 Message Register Low i.

Address: 0h base + 3BCh offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_REGION															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_REGION												CRCP			C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_PMRL*n* field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field. This bit is writeable in Config mode.
28–30 CRCP	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity. This bit is writeable in Config mode.
31 C	This bit will be set if CRC/P recalculation return an Error. This bit is writeable in Config mode.

56.3.106 PSI5 Message Register High i (PSI5_CH2_PMRH*n*)

The following figure shows the PSI5 Message Register High i.

Address: 0h base + 3C0h offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	O	F	EM	E	T	Slot_Counter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_PMRH*n* field descriptions

Field	Description
0 O	This bit carries the same information as OWSR, but appended here for each message separately. This bit is set when the current message (which is being read) has overwritten some previous unread message. Note that, if the corresponding location in the OWSR has been cleared or has been set by the SOWSR, then the same will be reflected in the message stored in the corresponding RAM location as well.
1 F	This represents that NO frame was received in corresponding configured Slot. This bit is writable in Config mode (for Test purposes). 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).

Table continues on the next page...

PSI5_CH2_PMRH*n* field descriptions (continued)

Field	Description
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel(M0, M1) i.e. at least one bit is absent during the Tbit period. This bit is writable in Config mode (for Test purposes). 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This Bit Indicates Electrical Error i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). This bit is writable in Config mode (for Test purposes). 0 No electrical error in fields other than M0, M1 and start fields. 1 Electrical error in fields other than M0, M1 and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot or it has spread across two slots or it has started before the first configured slot. This bit is writable in Config mode (for Test purposes). 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 Slot_Counter	This value indicates the slot number in which this frame was recieved. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,....Slot6. This bit is writable in Config mode (for Test purposes).
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH2_S*n*FCR) . In other words these bits contain either the start bit (S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appendded to the message is "0". It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM. This bit is writable in Config mode (for Test purposes).

56.3.107 SMC Frame Register n (PSI5_CH2_SFR*n*)

The following figure shows the SMC Frame Register n.

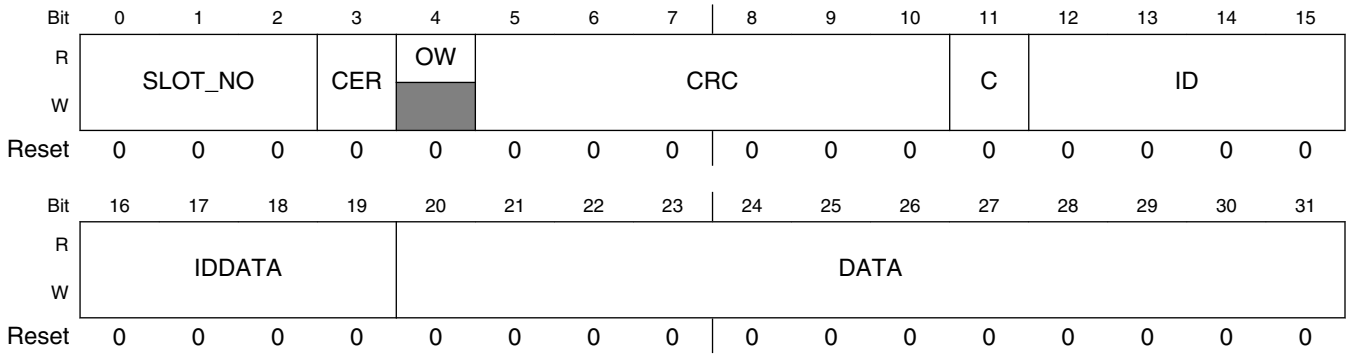
NOTE

Six dedicated 32-bit SFR[x] registers for SMC are available with a 1-1 slot correspondence. These can be read by either the CPU or the DMA (when DMA_EN_SF = 1).

NOTE

No FIFO for SMC frame is available. Six 32-bit registers are available in the address space, which can be accessed by either the DMA (using the DSFR) or the CPU (using the six SFR registers), depending on the setting of the DMA_EN_SMC bit.

Address: 0h base + 4BCh offset + (4d × i), where i=0d to 5d



PSI5_CH2_SFR*n* field descriptions

Field	Description
0–2 SLOT_NO	This indicates in which slot this SMC frame was received. Note that here the slot number refers to the slot in which all the re-arranged PSI5 messages (from which the M0,M1 bits have been extracted) are supposed to have occurred after the correct PSI5 message placement and Timing corrections have been performed. For SFR[1], slot_no = 1; For SFR[2], slot_no = 2; For SFR[3], slot_no = 3; For SFR[4], slot_no = 4; For SFR[5], slot_no = 5; For SFR[6], slot_no = 6; This field is writable in Config mode (for Test purposes).
3 CER	CRC Error This field is writable in Config mode (for Test purposes). 0 The present data has no CRC error. 1 The present data has a CRC error.
4 OW	Overwrite status **NOTE:** If the corresponding SOWSM bits in the SSESR have been set or the OWSM bits in the GISR have been cleared, then the same will be reflected in the message stored in the corresponding RAM location as well. The overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten, the overwrite bit gets set. 0 The present data was already read when new message arrived. 1 New message has overwritten the previous unread message.
5–10 CRC	6-bit CRC for slow serial message

Table continues on the next page...

PSI5_CH2_SFR*n* field descriptions (continued)

Field	Description
	This field is writable in Config mode (for Test purposes).
11 C	Configuration bit This bit is writable in Config mode (for Test purposes). 0 12-bit data and 8-bit message ID. 1 16-bit data and 4-bit message ID.
12–15 ID	Message ID: If C = '0' indicates ID[7:4], if C = '1' indicates ID[3:0]. This field is writable in Config mode (for Test purposes).
16–19 IDDATA	Message ID/DATA: If C = '0' indicates ID[3:0], if C = '1' indicates DATA[15:12]. This field is writable in Config mode (for Test purposes).
20–31 DATA	DATA payload This field is writable in Config mode (for Test purposes).

56.3.108 New Data Status Register (PSI5_CH2_NDSR)

The following figure shows the New Data Status Register.

Address: 0h base + 4D4h offset = 4D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NDS																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_NDSR field descriptions

Field	Description
0–31 NDS	New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when a new Message arrives i.e. faulty or non faulty in corresponding PSI5 MB location. Bit 0 reflects the MB0 location.These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SNDR.

56.3.109 Overwrite Status Register (PSI5_CH2_OWSR)

Address: 0h base + 4D8h offset = 4D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OWS																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_OWSR field descriptions

Field	Description
0–31 OWS	Over Write Status flags for PSI5 Messages corresponding to each PSI5 location in the RAM registers. These bits are set when any unread receive PSI5 message is overwritten by a newly arrived message). These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SOWSR.

56.3.110 Error Indication Status Register (PSI5_CH2_EISR)

The following figure shows the Error Indication Status Register.

Address: 0h base + 4DCh offset = 4DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERROR																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_EISR field descriptions

Field	Description
0–31 ERROR	Error Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when any of the five selectable (by ERROR_SELECT) bits (C, E, EM, T or F) is set for the corresponding PSI5 Message. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SEISR.

56.3.111 Set New Data Status Register (PSI5_CH2_SNDSR)

The following figure shows the Set New Data Status Register.

Address: 0h base + 4E0h offset = 4E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SNDS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_SNDSR field descriptions

Field	Description
0–31 SNDS	Sets New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. The corresponding NDS Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** This bit is writable in config and Normal modes.

56.3.112 Set Overwrite Status Register (PSI5_CH2_SOWSR)

The following figure shows the Set Overwrite Status Register.

Address: 0h base + 4E4h offset = 4E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SOWS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

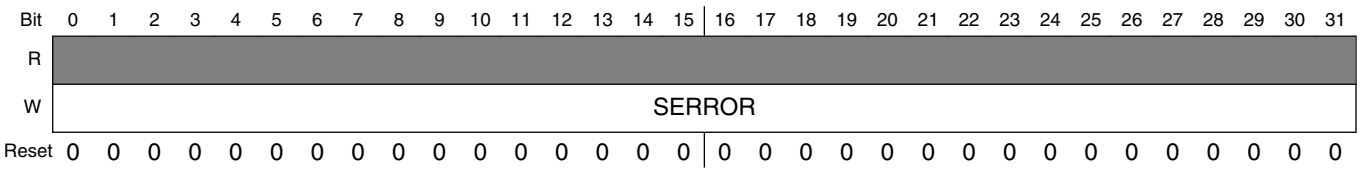
PSI5_CH2_SOWSR field descriptions

Field	Description
0–31 SOWS	Sets overwrite status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding OWS flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **NOTE:** These bits are writable in Config and Normal modes.

56.3.113 Set Error Status Register (PSI5_CH2_SEISR)

The following figure shows the Set Error Status Register.

Address: 0h base + 4E8h offset = 4E8h



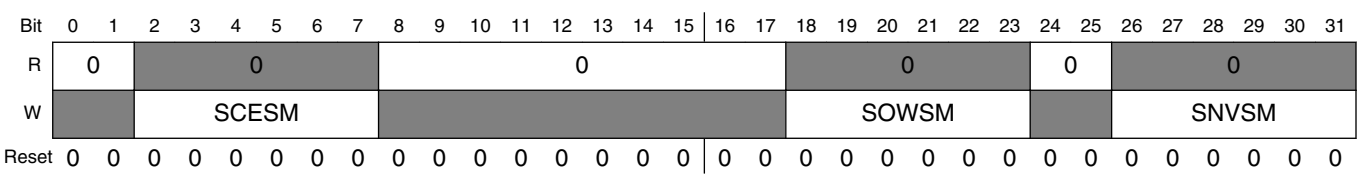
PSI5_CH2_SEISR field descriptions

Field	Description
0–31 SERROR	Sets Error Status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding ERROR flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **NOTE:** These bits are writable in Config and Normal modes.

56.3.114 Set SMC Error Status Register (PSI5_CH2_SSESR)

The following figure shows the Set SMC Error Status Register.

Address: 0h base + 4ECh offset = 4ECh



PSI5_CH2_SSESR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 SCESM	Sets IS_CESM Status flags for SMC Messages corresponding to each SMC MB locations.

Table continues on the next page...

PSI5_CH2_SSESR field descriptions (continued)

Field	Description
	The corresponding IS_CESM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** These bits are writable in Config and Normal modes.
8–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 SOWSM	Sets IS_OWSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_OWSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** These bits are writable in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SNVSM	Sets IS_NVSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_NVSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **NOTE:** This bit is writable in Config and Normal modes.

56.3.115 Sync Time Stamp Read Register (PSI5_CH2_STSRR)

The following figure shows the Sync Time Stamp Read Register.

Address: 0h base + 4F0h offset = 4F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								STSV																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_STSRR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 STSV	Sync Time Stamp Value[23:0] This bitfield provides the value of the free-running 24-bit time stamp counter triggered (captured) at last sync pulse.

56.3.116 Data Time Stamp Read Register (PSI5_CH2_DTSRR)

The following figure shows the Data Time Stamp Read Register.

NOTE

If S0 and S1 are detected in different slots then the Time Stamp value and the Slot Counter , both will correspond to the first slot(S0)

Address: 0h base + 4F4h offset = 4F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					SLOT_COUNTER			DTSV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DTSRR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SLOT_ COUNTER	3bit Slot Counter which corresponds to the slot for which the DTSRR contains the Time Stamp. The slot_counter contains the slot in which the start bits of this frame was captured (after successful detection of S1). If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. In case of asynchronous mode the slot counter has a fixed value of "1".
8–31 DTSV	Data Time Stamp Value[23:0] Whenever a valid sequence of the start bits (S0,S1) is detected then the value of the Time Stamp counter captured at the rising edge of the S0, is stored in this register at the rising edge of S1.

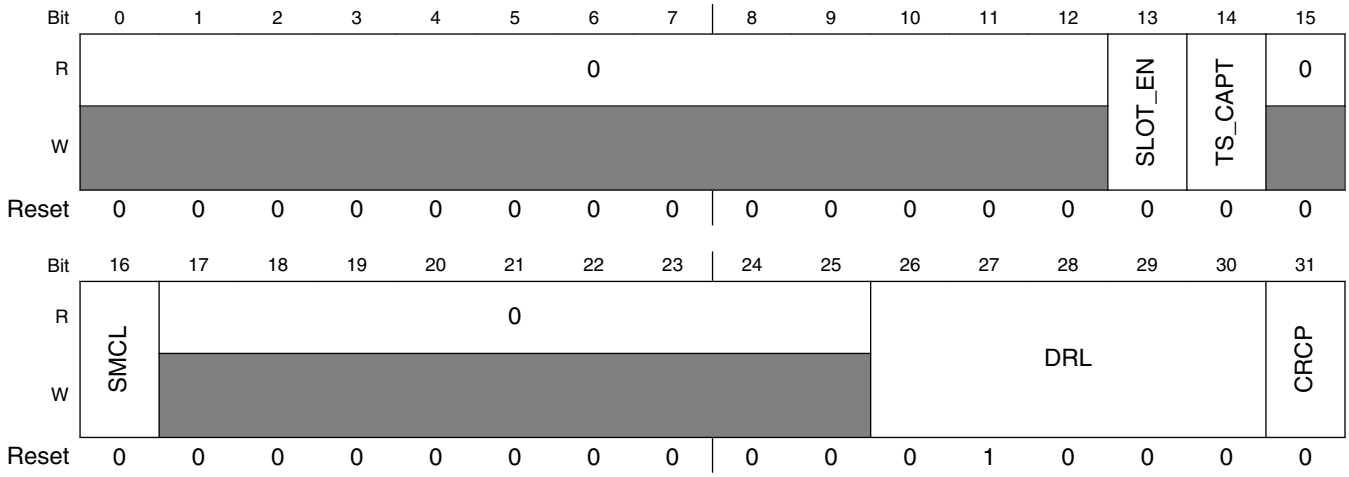
56.3.117 Slot n Frame Configuration Register (PSI5_CH2_S*n*FCR)

The following figure shows the Slot n Frame Configuration Register.

NOTE

In Asynchronous mode, the PSI5_S1FCR register has to be programmed for controlling the various parameters pertaining to the message received. The other registers PSI5_S[2-6]FCR have no effect in the asynchronous mode.

Address: 0h base + 4F8h offset + (4d × i), where i=0d to 5d



PSI5_CH2_S*n*FCR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SLOT_EN	SLOT_EN 0 This particular slot is disabled i.e not in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) have no effect. 1 This particular slot is enabled i.e is in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) are applicable.
14 TS_CAPT	TS_CAPT 0 Time stamp value captured at rising edge of S0 and stored Start sequence first edge. 1 Time stamp value captured at corresponding Tsync pulse (posedge) and stored with corresponding slot PSI5 message.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SMCL	Serial Messaging Channel field This field is used to enable the detection of start sequence of SMC frame on particular slot. Detection is not done if this field is '0'. 0 Serial Messaging Channel (optional) not present on Rx Message during Time Slot 'n' 1 2-bit{M1,M0} Serial Messaging Channel present on Rx Message during Time Slot 'n'
17–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–30 DRL	Data Region Length[4:0] This field represents Length of Data region (including Frame Control, Status , Data payload1&2 regions and optional messaging bits M0,M1) in Rx Message during Time Slot 'n' 00000-00111: Invalid 01000: 8 bit long data region 01001: 9 bit long data region 01010: 10 bit long data region ....

Table continues on the next page...

PSI5_CH2_S*n*FCR field descriptions (continued)

Field	Description
	11011: 27 bit long data region 11100: 28 bit long data region 11101-11111: Invalid
31 CRCP	CRCP 0 3 bit CRC[2:0] present on Rx Message during Time Slot 'n' 1 1 bit Parity field present on Rx Message during Time Slot 'n'

56.3.118 Slot 1 Start Boundary Register (PSI5_CH2_S1SBR)

The following figure shows the Slot 1 Start Boundary Register.

Address: 0h base + 510h offset = 510h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S1SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_S1SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S1SBT	Slot 1 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 1 should start.

56.3.119 Slot 2 Start Boundary Register (PSI5_CH2_S2SBR)

The following figure shows the Slot 2 Start Boundary Register.

Address: 0h base + 512h offset = 512h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S2SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_S2SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S2SBT	Slot 2 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 2 should start.

56.3.120 Slot 3 Start Boundary Register (PSI5_CH2_S3SBR)

The following figure shows the Slot 3 Start Boundary Register.

Address: 0h base + 514h offset = 514h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S3SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_S3SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S3SBT	Slot 3 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 3 should start.

56.3.121 Slot 4 Start Boundary Register (PSI5_CH2_S4SBR)

The following figure shows the Slot 4 Start Boundary Register.

Address: 0h base + 516h offset = 516h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S4SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_S4SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S4SBT	Slot 4 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 4 should start.

56.3.122 Slot 5 Start Boundary Register (PSI5_CH2_S5SBR)

The following figure shows the Slot 5 Start Boundary Register.

Address: 0h base + 518h offset = 518h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S5SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_S5SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S5SBT	Slot 5 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 5 should start.

56.3.123 Slot 6 Start Boundary Register (PSI5_CH2_S6SBR)

The following figure shows the Slot 6 Start Boundary Register.

Address: 0h base + 51Ah offset = 51Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S6SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_S6SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S6SBT	Slot 6 Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of the timing sync pulse, at which the slot 6 should start.

56.3.124 Slot n End Boundary Register (PSI5_CH2_SnEBR)

The following figure shows the Slot *n* Start Boundary Register (PSI5_S *n* SBR).

Slot n End Boundary register is a single register which can be programmed to define the "end" of the nth slot. The n can be defined in the SLOT_NO field of this register.

There are seven boundary registers for the six slots. Six of these registers (S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR) are used to define the start of each of the six slots. The seventh register SnEBR (Slot n End Boundary register) defines the end of the nth slot. The n is programmed in the slot_no field of this register.

The number of slots would automatically be available from the SnEBR. The SnEBR contains the slot number of the last slot and the end boundary of the same. Each slot can separately be enabled or disabled by using the slot_en bit in the SnFCR. When the start slot boundaries are properly defined in the SnSBR (Slot n Start Boundary Register) register but the slot_en bit = 0 in the corresponding SnFCR (Slot n Frame Configuration register) register, then the corresponding slot is treated as "available" but "unconfigured." For such a slot the data/diagnostic bits all remain 0. The "T" bit will, however, get set if a message that was supposed to come in a configured slot actually arrives in an unconfigured slot. Note that the various parameters like the Data length/CRC configuration and so on are valid ONLY for the configured slots. Even when sampling a message in an unconfigured slot, these parameters would be taken from the previous/next configured slot only. It is assumed that no data would be sent in an unconfigured slot. If a data is received in this unconfigured slot it would never be treated as belonging to the unconfigured slot. As a special case if "S0" and "S1" are detected in different slots then a T-bit error is set .

NOTE

The start slot boundaries in the S *n* SBR should be correctly programmed. If this is not done, the correct operation cannot be guaranteed. Slots occurring after the slot_no value programmed in the SnEBR are treated as being unavailable (that is, the data

in all the slots occurring after the SnEBR is rejected by the Manchester decoder).

Following are the examples:

- If only three slots are required between two sync pulses, program S1SBR, S2SBR, and S3SBR with the correct value of the start boundaries of the three slots and program slot_no = 3 in the S *n* EBR and define the end boundary of the third slot in S *n* EBR. The design ignores slots 4, 5, and 6.
- For six slots, the required programming would be: program the start of each of the six slots in S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Then program the end of the sixth slot in the S *n* EBR by putting slot_no = 6 and define the end boundary of the sixth slot in S *n* EBR.

Address: 0h base + 51Ch offset = 51Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													SLOT_NO		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SnEBT														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_SnEBR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 SLOT_NO	The field indicates the Slot number{1-6} for which Slot End Boundary Time is defined. Any value other than 1-6 is defaulted to 0. A value 0 here would mean that the manchester decoder would not sample any slots. This slot number also defines the number of slots that the IP assumes to exist between 2 sync pulses.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 SnEBT	Slot *n* End Boundary Time. This field specifies the time in intervals of 1µs, occurring after the rising edge of timing sync pulse, at which the slot 'n" should end.

56.3.125 Data Output Block Configuration Register (PSI5_CH2_DOBCR)

The following figure shows the Data Output Block Configuration Register.

Table 56-14. CMD_TYPE details

CMD_TYPE	PSI5 Frame [1]	Effect on registers	Writeable Register lengths
"000" - 0	Short Frame(V1.3) with 31 "1s" as the start condition	Command can be written to DPR for auto calculation of stuff/start/CRC bits. Can always be overwritten in DSR/DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	6 bits in DPR 15bits[2] in DBR/DSR.
"001" - 1	Short Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"010" - 2	Long Frame(V1.3) with 31 "1s" as the start condition	-do-	16bits in DPR. 29bits[2] in DBR/DSR.
"011" - 3	Long Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"100" - 4	X-Long Frame(V1.3) with 31 "1s" as the start condition	-do-	22bits in DPR. 37bits[2] in DBR/DSR
"101" - 5	X-Long Frame(V1.3) with 5 "0s" as the start condition.	-do-	-do-
"110" - 6	XX-Long (V2.0) -	-do-	24bits in DPR. 43bits[2] in DBR/DSR
"111" - 7	Non Standard Length.	Command can only be written to DSR and DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	Programmable from 1 to 64 bits in DBR/DSR, depending on the value of PSI5_DOBCR[DATA_LENGTH] field.

1. The start condition(31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command.
2. This is also the length of the command that will be shifted out during the ECU to Sensor communication, regardless of how many "actual" bits are written to DBR/DSR.

Table 56-15. DOBCR bit configurations and the related output states

Output path states	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL
State1	0	1	1
State2[1]	0	0	1
State3[1]	1	1	1
State4[1]	1	0	1
State5[1]	1	x	0

1. These states are shown in Valid states for integrated sync pulse generator. In State5, the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the "sdout" pin.

NOTE

Please refer to Figure 56-26 for the details about the above bits. Figure 56-26 is the actual implementation of Figure 56-25 such

that various contentions between the switches (S1, S2, and so on) do not happen.

Address: 0h base + 520h offset = 520h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA_LENGTH[5:0]						DBR_RST	DSR_RST	CMD_TYPE			DEFAULT_SYNC	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL	SW_READY
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DOBCR field descriptions

Field	Description
0–5 DATA_LENGTH[5:0]	Can take on values from 0 to 63 corresponding to 1bit to 64bit non standard length commands. When PSI5_DOBCR[CMD_TYPE] == 7(Non Standard) , then this field controls the length of command that can be written to the DBR and the DSR registers, else the value in this field is ignored. This when DATA_LENGTH = 0, then the length of the command that can be written to DSR and the DBR is 1, when DATA_LENGTH = 1, then the length of the command that can be written to DSR and the DBR is 2 and so on, until a maximum of 64 bit length command when DATA_LENGTH = 63. Note that the length of the command that is programmed by using this field, is also equal to the length of the command that shifts out of the DSR register. In other words, it is equal to the number of bits that shift out of DSR register once the DSR_RDY goes low . After these many shifts have happened , then the value corresponding to "DEFAULT_SYNC" starts getting shifted out and DSR_RDY goes as "1".
6 DBR_RST	This is to reset and reject current content to Data Buffer Register. When this bit is written as "1" then the contents of the DBR are reset to all "0s" (if DEFAULT_SYNC = = 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DBR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
7 DSR_RST	This is to reset and reject current content to Data Shift Register.When this bit is written as "1" then the contents of the DSR are reset to all "0s" (if DEFAULT_SYNC = = 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DSR would be ready for new data.Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
8–10 CMD_TYPE	These 3 bits indicate the type of command that needs to be transmitted during the ECU to sensor communication. Table 56-4 is a brief description of the same.
11 DEFAULT_SYNC	When this bit is set to "0" then the default value of DSR and DBR registers are all "0s"; when this bit is set to 1 then the default value of DBR and DSR registers are all "1s". This value is used as the value of the default sync pulses shifted at the output path when the DSR_RDY == 1.
12 GTM_TRIG_SEL	GTM event triggered/internal sync pulse generator selection as shift clock for the DSR. 0 Internal sync pulse generator shift triggered 1 GTM event shift triggered
13 SP_PULSE_SEL	Selects the source for the short pulse PWM module: 0 SP module gets the data from the DSR 1 SP module directly gets input from the GTM_event/internal sync pulse generator

Table continues on the next page...

PSI5_CH2_DOBCR field descriptions (continued)

Field	Description
14 OP_SEL	This bit selects as to which would be driving source of the "ipp_do_psi5_sdout" port. 0 The sync pulse generator select as per the Bit3 1 PWM output
15 SW_READY	When this bit is written to 1 the transfer from DBR to DSR automatically happens as soon as DSR_RDY becomes 1. When this bit is kept to 0 then the transfer from DBR to DSR will remain pending . Once the software makes this bit as "1" and DSR_RDY also goes as "1" the transfer to DSR takes place. However, note that when this bit is 0, and the DSR_RDY becomes 1, still the transfer to the DSR will NOT happen. Only when this bit is a 1 , will the transfer happen. This bit can be written in CONFIG and the Normal modes.

56.3.126 Manchestor Decoder Disable Offset (PSI5_CH2_MDDIS_OFF)

The Manchester Decoder Disable Offset register (PSI5_MDDIS_OFF) defines the time in intervals of 1 µs, for which the Manchester decoder remains disabled after the falling edge of the sync pulse. Using this register, this time can be programmed from 0 µs to 128 µs. This works on the 1 µs clock from the common clock generator module.

Address: 0h base + 522h offset = 522h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								MDDIS_OFF							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_MDDIS_OFF field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MDDIS_OFF	These 7 bits can be used to program the time for which the Manchester Decoder remains disabled. This offset is added AFTER the falling edge of the sync pulse. Thus the total time for which the manchester decoder remains disabled = TsyncH + MDDIS_OFF where TsyncH is the high time for the sync pulse . Thus TsyncH = Pulse_Width0 or Pulse_Width1 as the case maybe, depending on whether a "0" is being pulse modulated or a "1". Note that the Manchester Decoder will ALWAYS remain disabled during the high time of the Sync Pulse. Using this register it is possible to disable the Manchester Decoder BEYOND the Sync Pulse High Time as well. Figure 1032 shows the "MDDIS_OFF" definition.

56.3.127 Pulse Width for Data Bit Value 0 (PSI5_CH2_PW0D)

The following figure shows the Pulse Width for Data Bit 0 Register.

Address: 0h base + 524h offset = 524h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								Pulse_Width0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_PW0D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 Pulse_Width0	This defines the width (in µs) of data value '0' to be send from Data Output Register. It is the number of clock cycles(1µs clock) counted upto width "Pulse_Width0", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. When using PSI5 version 1.3 ECU-to-sensor Communication Pulse_Width0 should be configured to 0. Figure 1032 shows the definition of "Pulse_Width0".

56.3.128 Pulse Width for Data Bit Value 1 (PSI5_CH2_PW1D)

The following figure shows the Pulse Width for Data Bit 1 Register.

NOTE

When using the Version 1.3 format for ECU-to-sensor communication, the pulse width is NOT disabled. In this case, PW0D has to be programmed as 0. PW1D has to have the value of the length of the pulse desired when a 1 has to be transferred.

Address: 0h base + 526h offset = 526h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								Pulse_Width1							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_PW1D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH2_PW1D field descriptions (continued)

Field	Description
9–15 Pulse_Width1	This defines the width (in µs) of data value '1' to be send from Data Output Register. It is the number of clock cycles(1µs clock) counted upto width "Pulse_Width1", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. Figure 1032 shows the definition of "Pulse_Width1".

56.3.129 Counter Target Pulse Register (PSI5_CH2_CTPR)

The following figure shows the Counter Target Pulse Register.

NOTE

Please see Internal SYNC Pulse generation and coordination across PSI5 channels for details about the Sync Pulse generation and coordination.

Address: 0h base + 528h offset = 528h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CTPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_CTPR field descriptions

Field	Description
0–15 CTPR	Counter Target Pulse Register. This is the target counter value. Once the CTC (channel target counter) reaches the value of CTPR this CTC is reset. CTC is a 16 bit free running counter that starts/stops depending on the CTC_GED (in GCR) bit or the CTC_ED bit (in PCCR). It gets reset initially when its value reaches CIPR and subsequently when its value reaches CTPR. The fact that which out of CTC_GED or the CTC_ED bit enables/disables it, is dependent on the CTC_GED_SEL bit in the PCCR. When CTC_GED/CTC_ED == 0 then the CTC is reset. The minimum value of CTPR is 6. Values less than 6 will be defaulted to 0 and the CTPR counter will not start. The value in CTPR indicates the Time Period of the internally generated sync pulses. The un modulated (No PWM) internally generated sync pulses have a high time of 4-sp_ts_clk cycles i.e. the duty cycle of the un modulated internally generated sync pulses is (4/|CTPR| *100). The 4 sp_ts_clk cycles is just the nascent value of the Pulse which actually triggers the PWD counters Note that this gives one more possible state, in addition to the states shown in Valid states for integrated sync pulse generator . This additional state allows the output of the internal sync pulse generator to be made directly available at the output, without being pulse modulated. The CTPR describes ONLY the SYNC pulse period and not the sync pulse length which is governed by the PWD registers, except in State5 where in the GTM governs these parameters.

56.3.130 Counter Initialize Pulse Register (PSI5_CH2_CIPR)

The following figure shows the Counter Initialize Pulse Register.

NOTE

Due to internal Flop synchronisation there can be a jitter of +/- 20 ns between the offset times that are programmed in the CIPR of different channels and the ones actually being observed. Similarly there can be a jitter of +/- 20 ns in the programmed CTPR and the measured CTPR.

NOTE

Please see Internal SYNC Pulse generation and coordination across PSI5 channels for details about the Sync Pulse generation and coordination.

Address: 0h base + 52Ah offset = 52Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CIPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_CIPR field descriptions

Field	Description
0–15 CIPR	Counter initialize pulse register. This is the value using which the CTC gets reset, the first time after which it is enabled using CTC_GED (in GCR) or the CTC_ED bit (in PCCR). All subsequent resets are done when the CTC value reaches CTPR. The CIPR is used to set the offset time between the start of sync pulses between two different PSI5 channels, when the CTC of both of these channels are started simultaneously using CTC_GED. The offset between any two PSI5 channels would be |CIPR_channel1 - CIPR_channel2| sp_ts_clk cycles. This if channel 1 has a CIPR = 0 while channel2 has CIPR = 10, then the offset would be 10 sp_ts_clk cycles.

56.3.131 Data Preparation Register Low (PSI5_CH2_DPRL)

The following figure shows the Data Preparation Register Low.

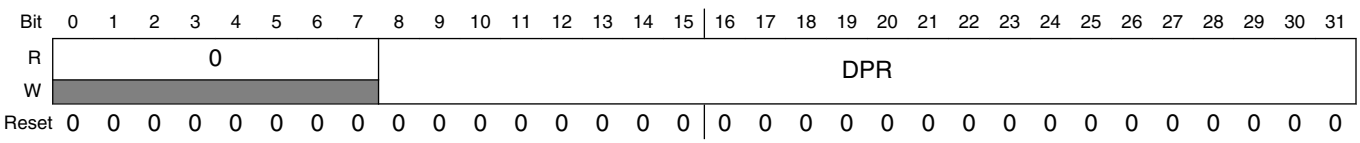
This register is the lower register used for writing the standard data and the address commands and is to be used for writing standard length ECU-to-sensor commands. The CRC/stuff/start bits are automatically appended by the hardware before the command is transferred to DBR. The commands to this register are accepted ONLY when PSI5_GISR[DPR_RDY] == 1. Whenever PSI5_GISR[DPR_RDY] == 1 , then each bit of

this register defaults to 0 . Writing to this register when PSI5_GISR[DPR_RDY] == 0 will cause the PSI5_GISR[IS_PROW] status bit to be set and the generation of an error interrupt, if it is enabled. Further , trying to write to this register when PSI5_DOBCR[CMD_TYPE] = "7" results in the setting of the PSI5_GISR[IS_PROW] bit and the resulting write is rejected.

Table 56-16. DPR details

PSI5_DOBCR[CMD_TYPE]	Standard PSI5 FRAME Types	Writeable bits in PSI5_DPRL
"0" or "1"	Short Frame(PSI5 V1.3)	6(DPR[5:0])
"2" or "3"	Long Frame(PSI5 V1.3)	16(DPR[15:0])
"4" or "5"	X-Long Frame(PSI5 V1.3)	22(DPR[21:0])
"6"	XX-Long Frame(PSI5 V2.0)	24(DPR[23:0])
"7"	Not Available for write	Not Available for write

Address: 0h base + 52Ch offset = 52Ch

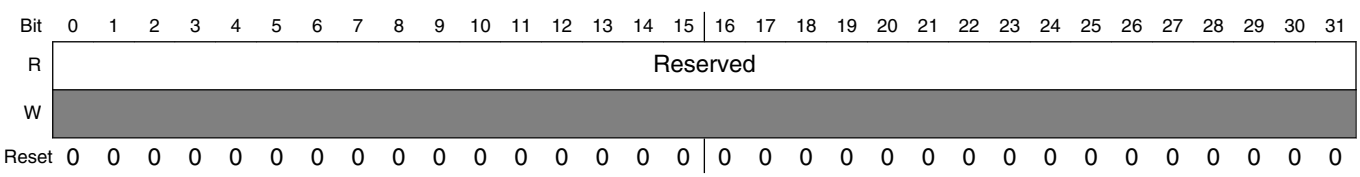


PSI5_CH2_DPRL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 DPR	DPR[23:0] are the 24-bits of the DPR register used for writing the variable length , standard ECU-to-Sensor commands , comprising of the address , data and other fields . Note that the IP is transparent to the arrangement of the address,data and other fields and it treats the bits in these fields identically for the purpose of CRC calculation and transmission. This register is writeable only when using the standard length PSI5 frames . For details about CRC calculation please refer to Data transmission . The writeable bits in this register are governed by the PSI5_DOBCR[CMD_TYPE] bit fields as per Table 56-6 . Note that the unwriteable bits in this register field are always read as "0s" . This register has to be written by the CPU. These bits can be written only in the Normal mode.

56.3.132 Data Preparation Register High (PSI5_CH2_DPRH)

Address: 0h base + 530h offset = 530h



PSI5_CH2_DPRH field descriptions

Field	Description
0–31 Reserved	This field is reserved. This register is always read as 0s. Writing to this register does not have any effect.

56.3.133 Data Buffer Register Low (PSI5_CH2_DBRL)

The following figure shows the Data Buffer Register Low.

This register is writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode.

NOTE

Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1, then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 56-17. DBRL[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" (Non Standard Length)	DBR[31:0]]	Variable length of writeable bits , as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
"6"(XX Long)	DBR[31:30]	DPR[19:18]
	DBR[29]	1'b0(stuff)
	DBR[28:23]	DPR[17:12]
	DBR[22]	1'b0(stuff)
	DBR[21:16	DPR[11:6]
	DBR[15]	1'b0(stuff)
	DBR[14:9]	DPR[5:0
	DBR[8:0]	"0111111110"

Table continues on the next page...

Table 56-17. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"4" or "5" (X-Long)	DBR[31]	1'b1(stuff)
	DBR[30:28]	DPR[20:18]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	DPR[17:15]
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"2" or "3" (Long)	DBR[31:29]	"000"
	DBR[28]	CRC[0]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	{CRC[1],CRC[2],DPR[15]}
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"0" or "1" (Short)	DBR[31:15]	0
	DBR[14:12]	{CRC[0],CRC[1],CRC[2]}[
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]

Table continues on the next page...

Table 56-17. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)

Address: 0h base + 534h offset = 534h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DBRL field descriptions

Field	Description
0–31 DBR	This register contains the Lower 32 bits(DBR[31:0]) of the max-64 bit length command (DBR[63:0]). The higher bits , DBR[63:32] are contained in the DBRH register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 56-7 . The table (except when PSI5_DOBCR[CMD_TYPE] = "7") show the bit assignment of the DBR[31:0] when these bits are automatically updated by the hardware. When PSI5_DOBCR[CMD_TYPE] == "7"), then only the CPU can write to these registers.

56.3.134 Data Buffer Register High (PSI5_CH2_DBRH)

The following figure shows the Data Buffer Register High.

This register is CPU writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY]== 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit

PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 56-18. DBRH[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" - Non Standard Frame lengths	DBR[63:32]]	Variable length of writeable bits , as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
"6" - XX Long	DBR[63:43]	All 0s
	DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1], CRC[0]}
	DBR[36]	1'b0(stuff)
	DBR[32:35]	DPR[23:20]
"4" or "5" -X Long	DBR[63:37]	All 0s
	DBR[36]	CRC[0]
	DBR[35]	1'b1((stuff)
	DBR[34:32]	{CRC[1],CRC[2],DPR[21]}
"3" or "2"(Long) OR "1" or "0" (Short)	DBR[63:32]	All 0s

Address: 0h base + 538h offset = 538h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DBRH field descriptions

Field	Description
0–31 DBR	This register contains the Upper 32 bits(DBR[63:32]) of the max-64 bit length command (DBR[63:0]). The lower bits, DBR[31:0] are contained in the DBRL register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 56-8 . Note that when PSI5_DOBCR[CMD_TYPE] == "7") , then only the CPU can write to these registers.

56.3.135 Data Shift Register Low (PSI5_CH2_DSRL)

The following figure shows the Data Shift Register Low.

This register is CPU writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addtion to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 53Ch offset = 53Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DSRL field descriptions

Field	Description
0–31 DSR	This register contains the lower 32 bits, (DSR[31:0]), of the max-64 bit length command (DSR[63:0]). The higher bits, DSR[63:32], are contained in the DSRH register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits.When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRL . Thus when updated by the hardware, then DSR[31:0] = DBR[31:0] . Hardware updation occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0 then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1 . These bits can be written by the CPU in the Normal mode , when PSI5_GISR[DSR_RDY] == 1'b1 .

56.3.136 Data Shift Register High (PSI5_CH2_DSRH)

The following figure shows the Data Shift Register High.

This register is writable only in Normal mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addtion to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 540h offset = 540h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH2_DSRH field descriptions

Field	Description
0–31 DSR	This register contains the higher 32 bits (DSR[63:32) of the max-64 bit length command (DSR[63:0]). The lower bits, DSR[31:0], are contained in the DSRL register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRH . Thus when updated by the hardware, then DSR[63:32]] = DBR[63:32] . Hardware updation occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0 then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1 . These bits can be written by the CPU in the Normal mode , when PSI5_GISR[DSR_RDY] == 1'b1 .

56.3.137 Device modes and Register bit accesses

1. The following bits are writable in all modes (Normal, Config, and Disable).
   - GCR[GLOBAL_DISABLE_REQ]
   - PCCR[PSI5_CH_CONFIG]
   - PCCR[PSI5_CH_EN]
   - Global Counter Enable/Disable (GCR[CTC_GED]). This is to allow the Sync Pulse Generator to be enabled/disabled independently among those channels that are used for staggering, irrespective of the state of the channels NOT being used for staggering.

2. The following register is writeable only in Normal mode and is initialized to all 0s.
   - DPRL
3. The following registers are writable ONLY in Normal mode. Initialization in Config mode depends on the value of PSI5_DOBCR[DEFAULT_SYNC] bit. When PSI5_DOBCR[DEFAULT_SYNC] == 1 in Config mode, then the below registers are initialized to all 1s in Normal mode. When PSI5_DOBCR[DEFAULT_SYNC] = 0 in Config mode, then the below registers are initialized to all 0s in Normal mode. Once in Normal mode, these registers can then be programmed for any command that needs to be sent. x
   - DBRH/DBRL
   - DSRH/DSRL
4. The following registers are writable in Config and Normal modes only:
   - All interrupt control registers
   - All SET registers (the one shot registers which are used to set the error flags for software drivers)
   - DMA Control register
   - Channel Enable/Disable (PCCR[CTC_ED]). Using this it is possible to allow the Sync Pulse Generator to be enabled/disabled in Normal mode. The counter is reset when CTC_ED == 0
   - All w1c bits
5. Apart from these, all other register bits can be written only in Config mode , including the Data registers(PMRH/PMRL/SFRn), that, in Normal mode, are updated by the hardware only.
6. Debug mode: If the device is in Normal mode and the debug mode is enabled, then ONLY the registers mentioned in pts. 1, 2, 3, and 4 above can be written by the debugger. Register bits mentioned in pt. 5 above, will not be available for write through the debugger. In case the debugger has to change any register, then the device has to be brought in the Disable -> Config mode. The debugger can, however, read any register in any mode.

56.4 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. The following figure shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

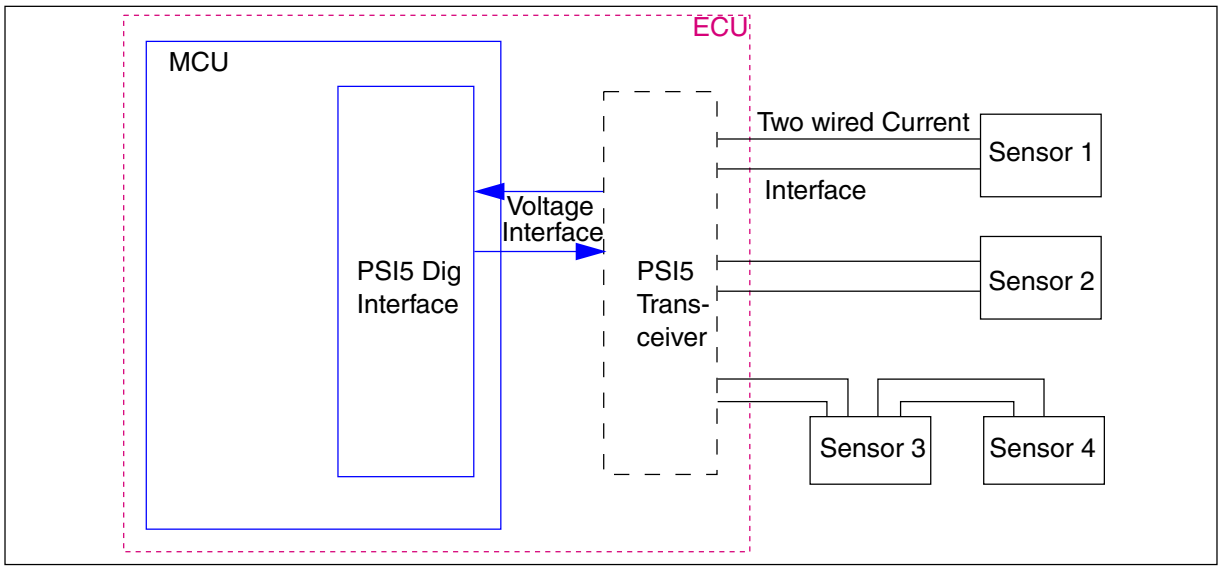


Figure 56-3. System setup (peripheral sensor connected to device)

The ECU (Electronic Control Unit) comprises PSI5 digital interface and the PSI5 transceiver (present externally).

56.4.1 Sensor-to-ECU communication

56.4.1.1 Physical layer

The PSI5 uses two wires to provide both power supply and data transmission to the sensors. The ECU provides a pre-regulated voltage to the sensors. Data transmission from the sensor to the ECU is done by current modulation on the power supply lines. Current oscillations are damped by the ECU.

56.4.1.2 Bit encoding

A "low" level ($I_{S,Low}$) is represented by the Normal (quiescent) current consumption of the sensor(s). A "high" level ($I_{S,High}$) is generated by an increased current sink of the sensor ($I_{S,Low}$ + …I_S). The current modulation is detected within the PSI5 transceiver.

Manchester coding is used for data transmission. A logic 0 is represented by a rising slope and a logic 1 by a falling slope of the current in the middle of T bit.

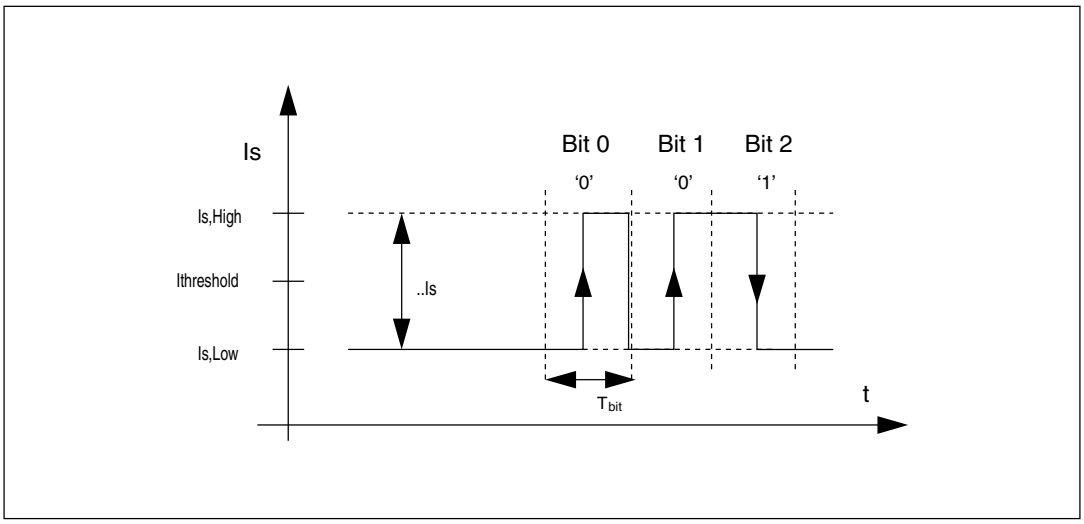


Figure 56-4. Bit encoding of data from sensor

56.4.1.3 Synchronization

For synchronized operation and bus mode, a sync pulse is generated by the PSI5 receiver. The sync pulse is output from the integrated PSI5 receiver to an external PSI5 transceiver that processes these pulses to a pre-regulated voltage level on supply lines to sensors. The voltage change is detected within the sensors.

56.4.2 Data Link Layer

The PSI5 message is received as serial Manchester-coded bits. The bit stream is passed through a Manchester decoder stage that converts it to normal NRZ format bit by bit and stores it serially in the receive register (64 bits). This block clearly identifies start (00) and end of receive frame using preprogrammed frame configuration registers.

- The Manchester-coded bits can be received at the rate of 125 Kbit/s.
- The Manchester decoder is designed to consider data with duty cycle of 30–70%.
- The bit decoding is done using an oversampling and counting algorithm with majority voting and edge recognition. Clocks of (125 kHz) × 32 (4 MHz also referred to as 32× clock) are supplied from a common clock generator module. This clock is used for oversampling of the receive bit and thereby decoding it as normal 1 or 0 (NRZ).

- The Manchester decoder is masked (switched off) for the duration when the sync pulse is transmitted. In addition to this, it can be configured to be off for an additional time, after the falling edge of the sync pulse. The duration of this time is configurable in the MDDIS_OFF register.
- It uses a 2oo3 voter to recognize the encoded half bits.
- There is a 3oo5 majority voting glitch filter at the data input of the Manchester decoder.

56.4.2.1 Reception of data frames

The extended PSI5 (Power Train Applications) receive data frame consists of *p* bits containing:

- Two start bits (S1 and S2) {S1=0, S2=0}
- One parity bit (P) with even parity (or 3 CRC bits C0, C1, and C2).
- A data region (D0…..D[*k* – 1]) with *k* = 8 to 28 bits with 1-bit granularity.
- The total length of a PSI5 frame is p = *k* + 3 data bits (in case of frames with parity bit) or p = *k* + 5 data bits (in case of frames with CRC).
- Data bits are received LSB first. The parity or CRC check bits cover the bits of the entire data region.

The length of the data region can vary between *k* = 8 to 28 bits (with 1-bit granularity). The data region can be split into the following fields and regions:

- The bits D0 and D1 can optionally represent Serial Messaging Channel M0, M1.
  - This Serial Messaging Channel (SMC) is used to receive initialization data during fast startup phase or slow acknowledgement/response messages (to command transmitted from ECU requesting data or diagnostic information from the sensor).

Note, that the maximum length of the entire data region is 28 bits. Thus the signal payload region and the optional data fields cannot concurrently be used at their maximum length in some cases.

As the PSI5 module is targeted for powertrain applications, the data frame format differs from standard PSI5 frames for safety applications specified in version 1.3 of the PSI5 specifications. More specifically for supporting the version 2.0 powertrain substandard. The Powertrain data frame format and optional Serial Messaging Channel (SMC) frame format are shown in the following figure.

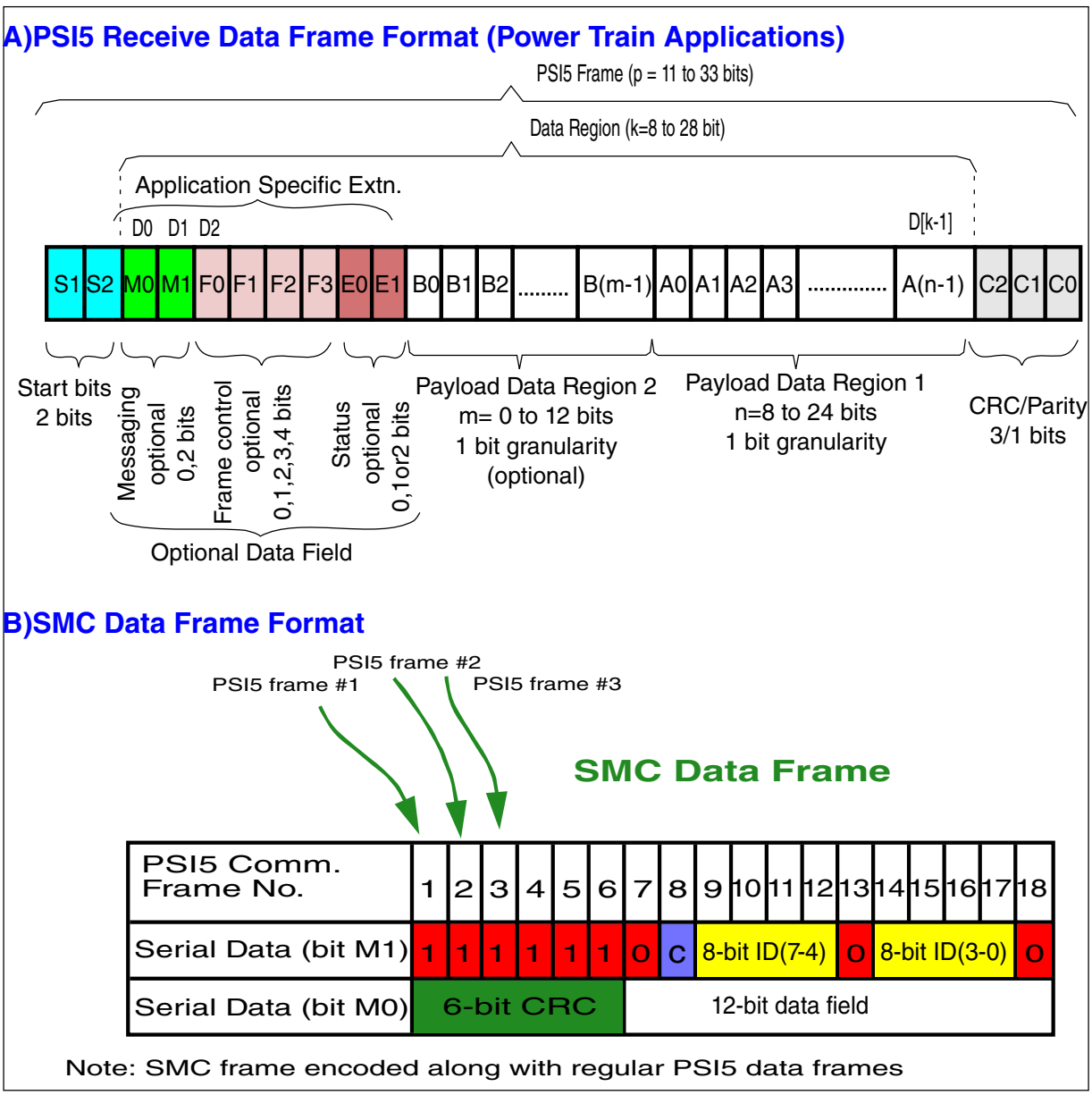


Figure 56-5. Receive data frame format

56.4.2.2 Data Frame Configuration Control

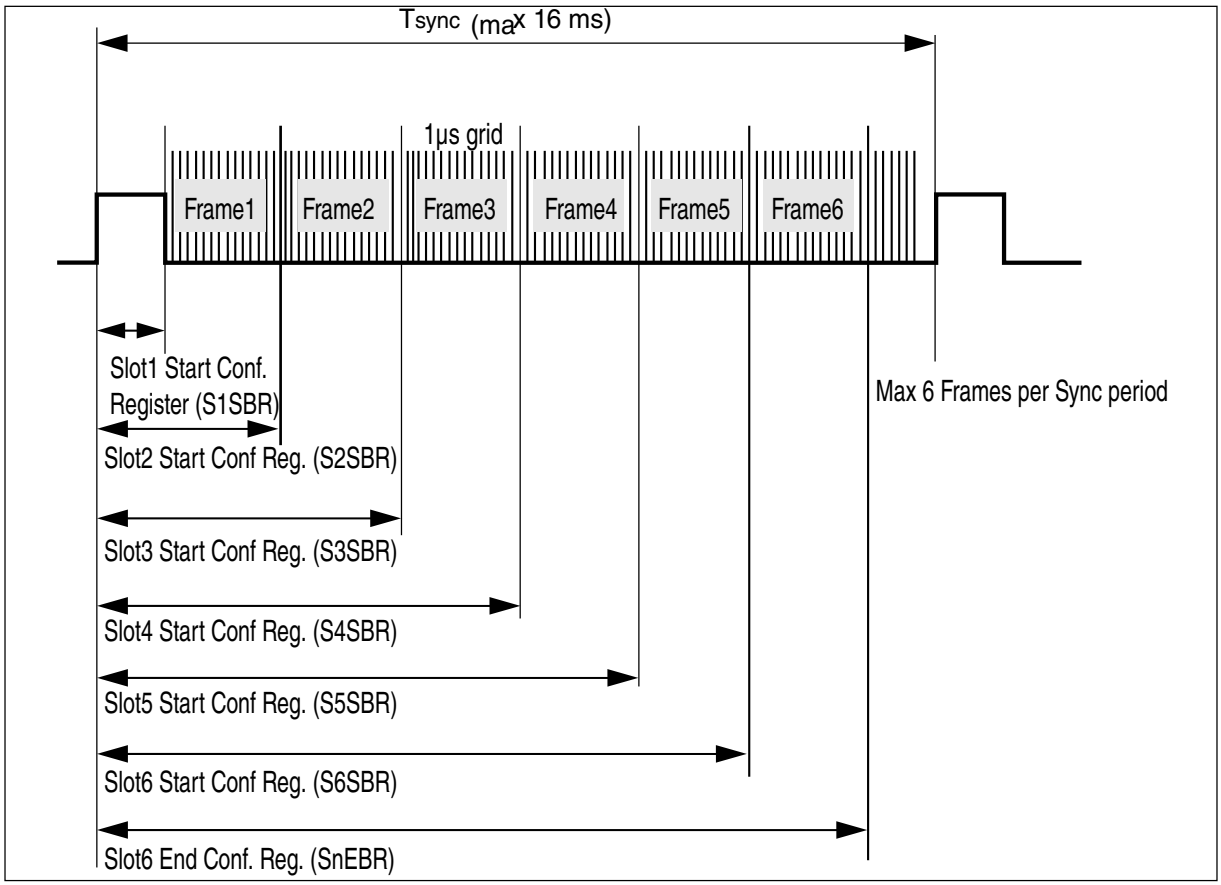


Figure 56-6. Data frame configuration and sampling

At the receive path, as many as six frame slots (reception windows) can be programmed with a start time of consecutive slots (slot1 to slot6) and end time or last slot which can be configured through the Slot *n* Start/End Configuration Registers (Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)-Slot 6 Start Boundary Register (PSI5_CH0_S6SBR) and Slot n End Boundary Register (PSI5_CH0_SnEBR)) there are seven registers for six slots. The slot time value is with respect to timing sync pulse.

The following range is always maintained for T_{sync} (during programming of the GCR and CTPR registers) when timing (sync) pulses are generated through the Sync Pulse Generator.

$$T_{sync} < 16\text{ms}$$

For the maximum defined six frame slots, six frame configuration registers (S1FCR, S2FCR, S3FCR, S4FCR, S5FCR, and S6FCR) are also available. In these registers it is possible to configure the payload region of the expected frames, in addition to other channel-specific parameters.

56.4.2.3 Data frame handling

The following figure shows the detection of the start bits and the various parameters for the same.

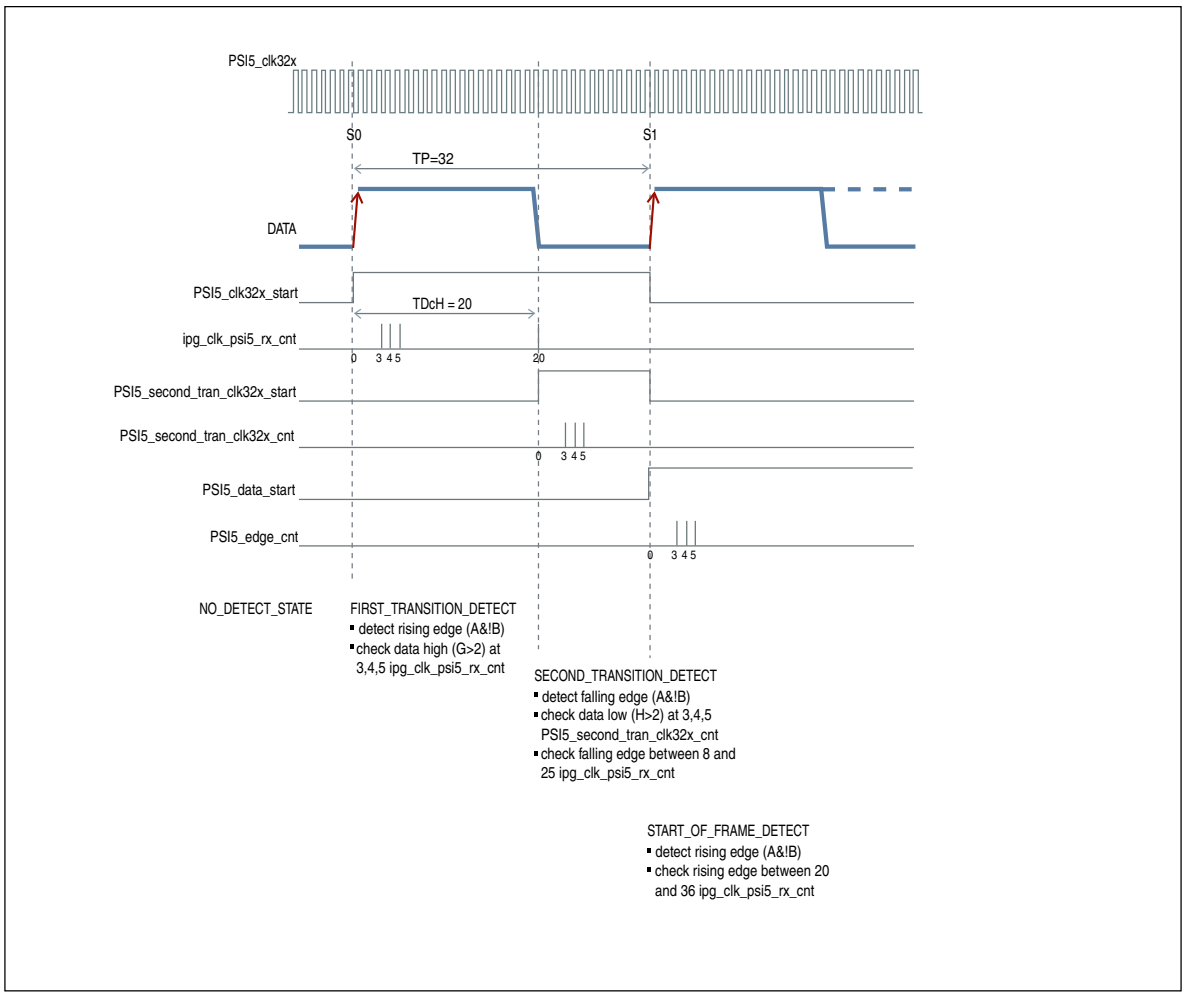
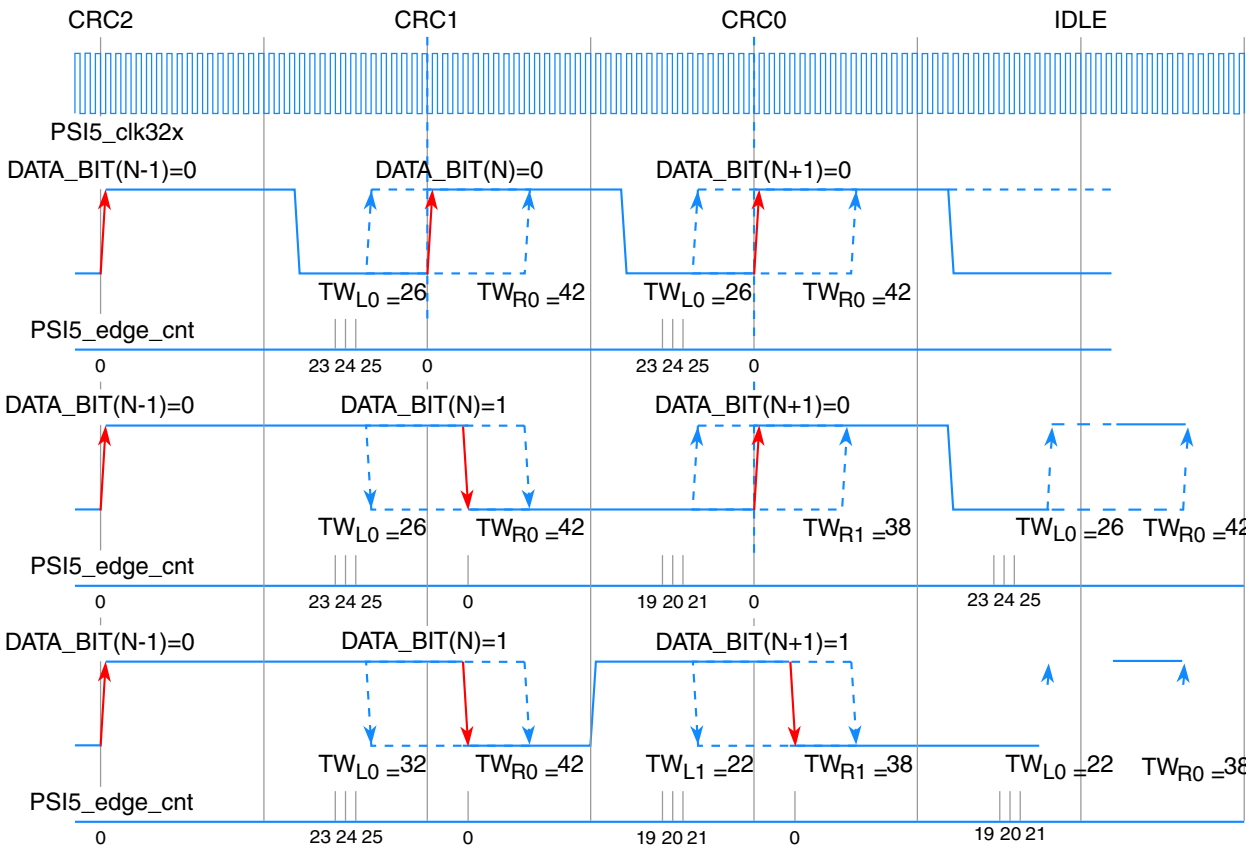


Figure 56-7. Start detection

Figure 56-8. Start Bit Detection

The following figure shows how the bits are differentiated into a “1” or a “0” based on the various margins and sample points.



Start Sequence Measured: Tp = 32 and TDcH = 20
Calculations:
TDcL = Tp–TDcH=12, DeltaDc = TDcH–Tp/2 = 4; TMarginL = TDcL/2=6; TMarginH = TDcH/2 = 10
If DATA_BIT(N) = 0, TWI = TDcH+TMarginL=26; signal sampled at TWI-3, -2 and -1

- If signal low, expect rising edge between TWL0 and TWR0 then DATA_BIT(N+1) = 0.
  TW_{L0} = TDcH+TMarginL and TWR0 = Tp+TMarginH
- If signal high, expect falling edge between TWL0 and TWR0 then DATA_BIT(N+1) = 1.
  TW_{L0} = TDcH+TMarginL and TWR0 = Tp+DeltaDc+TMarginL

If DATA_BIT(N) = 1, TWI = TDcL+TMarginH = 22; signal sampled at TWI-3, -2 and -1

- If signal low, expect rising edge between TWL1 and TWR1 then DATA_BIT(N+1) = 0.
  TW_{L1} = TDcL+TMarginH and TW_{R1} = Tp–DeltaDc+TMarginH
- If signal high, expect falling edge between TWL1 and TWR1 then DATA_BIT(N+1) = 1.
  TW_{L1} = TDcL+TMarginH and TW_{R1} = Tp+TMarginL

IDLE state can be reached in two ways:

1. Idle can be detected after zero(last data bit) .In this the edge is searched in the same window as for data zero i.e. TWLO-26 to TWRO = 42
2. Idle is detected after one(last data bit).In this the edge is searched in the same window as for data one i.e. TWLO = 22 to TWRO = 38

Figure 56-9. Bit Extraction

56.4.2.3.1 Start of frame recognition

56.4.2.3.2 Data state

In the data state, the Manchester decoder samples the number of data bits corresponding to the configured length of the data payload region and 3-bit CRC or 1-bit parity.

56.4.2.3.3 Message diagnostic checks

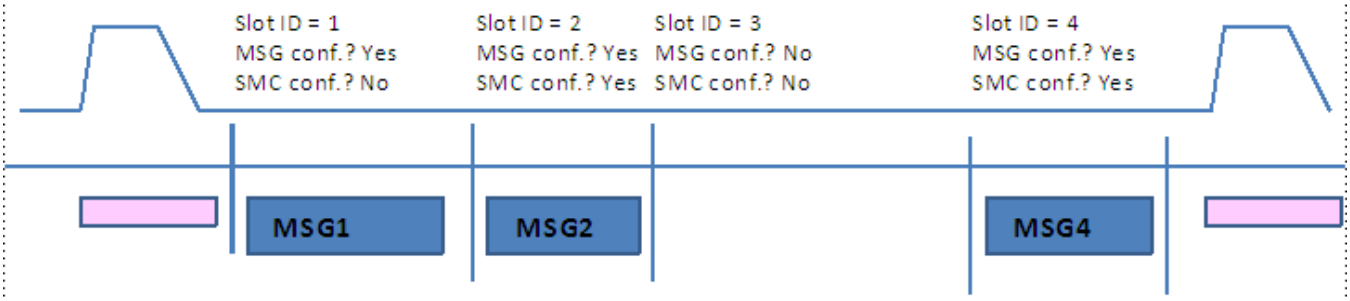
During the data state, electrical correctness of each bit in regard of Manchester encoding is checked. If an incorrect bit is detected, a 0 is written into the data payload region. Following completion of reception, correctness of the entire message is checked with the CRC and timing of the message reception.

- CRC is recalculated for the data and compared with received CRC bits. The CRC comparison result is stored as a status bit, denoted as the C field in the receive register and the RAM register. Irrespective of the value of the “C” bit the, the frame is always stored in the RAM register and is never discarded.
- If at least one bit is electrically faulty (apart from M0M1), then the bit field E is set to 1 and stored as status bit in the receive register and the RAM register.
- The EM bit is set to 1 if any of M0 or M1 bits are electrically faulty. Note that the E and the EM bits can be set only when a valid start sequence has been detected.
- Set status bit field T in the receive register to 1 if the frame received is in the wrong slot (boundary violations). This bit is NOT set for asynchronous modes.
- Set status bit F if no frame is received in defined frame slot. This bit is NOT set for asynchronous modes.
- The incoming frames after the sync pulse are to be numbered from 1 to 6 (max). This is done by a 3-bit slot counter, which is incremented automatically after the end of each slot irrespective of whether a valid message is received or not. This 3-bit value is stored in the receive register and the RAM registers as the "slot_counter" field.

56.4.2.3.4 T-bit handling

The following paragraphs describe how the IP handles error cases that generate T bit errors. A number of examples are shown, including the error free case, which is used to explain the reception process.

56.4.2.3.4.1 Error free case



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

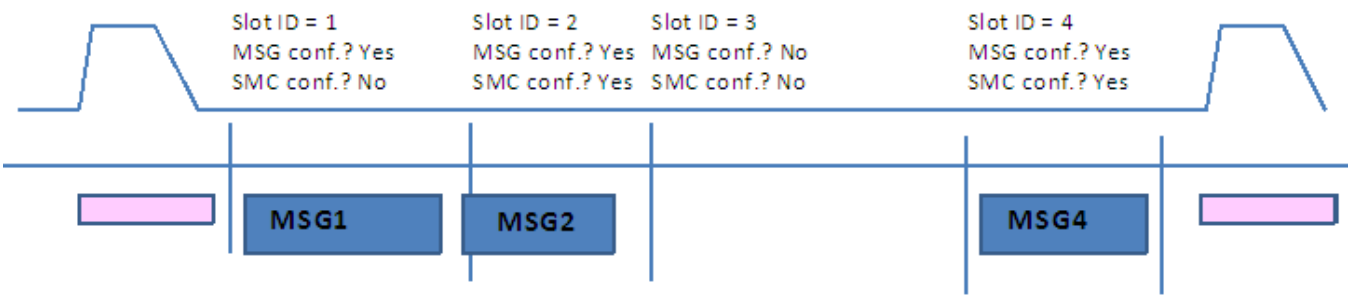
Figure 56-10. Error free case

In this case, all messages are received in the correct slots.

- As soon as the Manchester decoder block is enabled (after the SYNC pulse has finished and any configured disable offset has expired, pink region) the IP checks for the first configured message and starts monitoring the output of the PSI5 transceiver.
- The IP identifies a correct start of message sequence (S0, S1)== ’00’, and captures the slot counter when the first rising edge of the start sequence is detected. This first rising edge detected is the mid-point transition of the Manchester encoded S0 bit of the message start sequence.
- The expected number of bits configured in the message is received and decoded. Any error in the Manchester encoded bits of the message E bit is set - unless the error is in one of the first two bits and SMC message is configured, in which case the EM bit is set.
- After the final bit is received, a check is made that the message CRC or parity is correct. If the CRC or parity check is not correct, C bit is set for the message.
- Once the CRC checks have been made, the received message — along with the status bits, slot counter in which the start sequence was received, the timestamp of the message, the message data and CRC or parity is saved to the channel message FIFO.

- As soon as the received message has been stored,the idle state of the bus is verified and monitoring of the transceiver output restarts to check for the next configured message.
- If the serial message channel (SMC) is configured, the first two data bits are extracted from a sequence of 7 PSI5 messages as long as no electrical error is detected in those bits (EM error). If the correct start sequence for the SMC message is detected on M1 (6 logical 1s and a logical 0) in those 7 messages, M0 and M1 from the following 11 messages are also extracted and saved in the SMC buffer allocated to the slot in which the PSI5 message is received (in total, 18 PSI5 messages are needed to build one SMC message). When the SMC has been received, the CRC is checked and the status of the check is saved in the SMC buffer. If an EM error occurs after a start sequence has been detected and before the end of the message is reached, M0 and M1 are both saved as 0 and the rest of the message is received, at which point the CRC is checked as normal.

56.4.2.3.4.2 Message Too Early



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	1	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

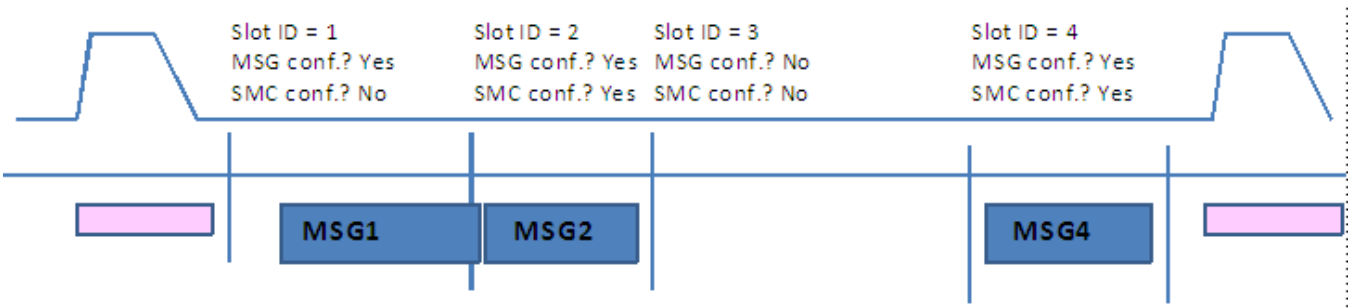
Figure 56-11. Message Too Early

In this scenario, MSG2 reception starts early, before slot 2 boundary is crossed, and so the slot counter is captured as “1” and because the message starts in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the

correct MSG2 has been received but not entirely in it's correct slot. The status of the message will include the T bit set and slot counter will be 1, so software can interpret that MSG2 was too early.

The SMC message is received correctly.

56.4.2.3.4.3 Message Too Late



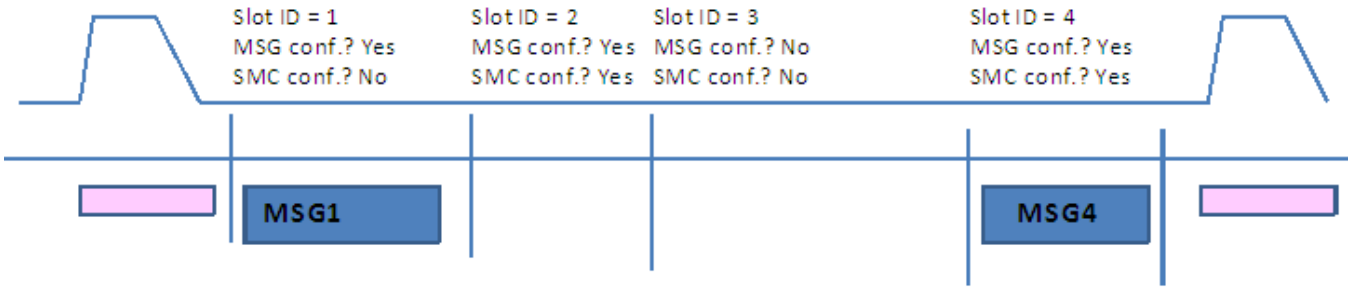
Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	Yes	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

Figure 56-12. Message Too Late

In this scenario, MSG1 reception starts later than expected and ends after slot 2 boundary is crossed. The slot counter is captured as "1" and because the message ends in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the correct MSG1 has been received but not entirely in it's correct slot. The status of the message will include the T bit set and slot counter will be "1", so software can interpret that MSG1 was late.

The SMC message is received correctly.

56.4.2.3.4.4 Message Missing



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	Yes	No	No	2	0/0
MSG4	4	No	No	No	No	4	M0/M1

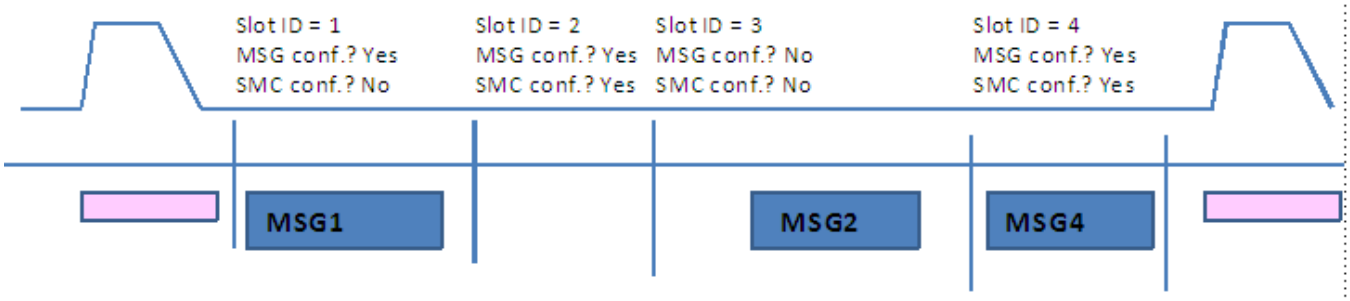
Figure 56-13. Message Missing

In this scenario, MSG2 is missing.

At the end of slot 2, the IP writes message buffer location 2 (MSG2 location) with slot counter 2 and the F bit set. The data and timestamp regions are filled with 0s.

The IP waits until the start of the next configured slot (slot 4) to update the SMC buffer 2 with “00”. If slot 4 was not configured, the SMC for the missing MSG2 would be updated at the end of the final slot.

56.4.2.3.4.5 Message Too Much Late (in unconfigured slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2a	2	No	Yes	No	No	2	No update
MSG2b	3	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

Figure 56-14. Message Much Too Late (in unconfigured slot)

In this scenario, MSG2 reception doesn't start in it's own slot, and the F bit is set for message buffer location 2 (MSG2a).

Because slot 3 is not configured, the IP continues to wait for MSG2 in slot 3 and does not update the SMC message buffer 2. The IP recognizes the MSG2 start condition in the normal way and updates PSI5 message buffer 3 (MSG2b), capturing slot 3 as the start slot counter and setting T bit, so that the software can recognize that MSG2 has come in slot 3.

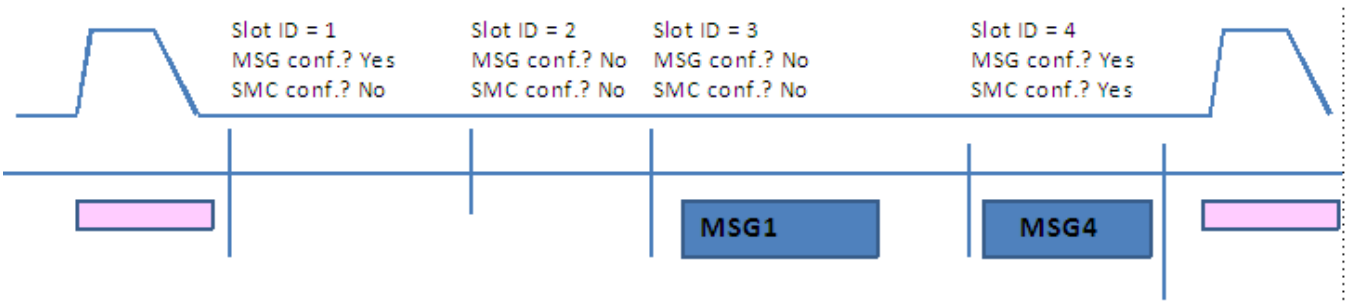
When MSG2 has been received, SMC message buffer 2 is updated at the end of slot 3. MSG4 is received normally.

NOTE

Any scenario where an expected message arrives in an unconfigured slot, leads to an issue when DMA access is used to read the PSI5 message buffer. This is because instead of the expected three entries in the PSI5 message buffer, there will be four, and assuming the DMA watermark is set for the expected three messages, the DMA request will be activated before MSG4 has been received.

In order to overcome this, if DMA access is required, no slots can be left unconfigured and the DMA watermark must be set to the number of slots in the SYNC period. In that way, the DMA will always be syncronised, but depending on how the 'empty' slot is configured, when a message is too late as described here, the result can be that the late message and its associated SMC content is lost.

56.4.2.3.4.6 Message Much Too Late (in unconfigured slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1a	1	No	Yes	No	No	No	No
MSG1b	3	Yes	No	No	No	No	No
MSG4	4	No	No	No	No	4	M0/M1

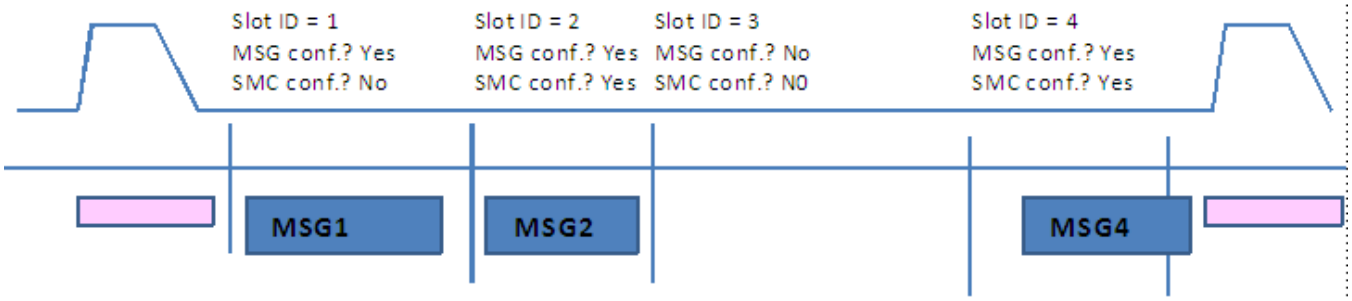
Figure 56-15. Message Much Too Late (in unconfigured slot)

In this scenario, MSG1 is received much too late, in the second of two unconfigured slots.

When MSG1 does not arrive in slot 1, MSG1a is saved in the message buffer with F bit set. The IP waits throughout the two unconfigured slots and is received in slot 3 with slot counter 3. Because MSG1b is received completely in slot 3 it is saved in the message buffer with T bit set.

MSG4 is received normally.

56.4.2.3.4.7 Message Too Late (in last slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	Yes	No	No	No	4	M0/M1

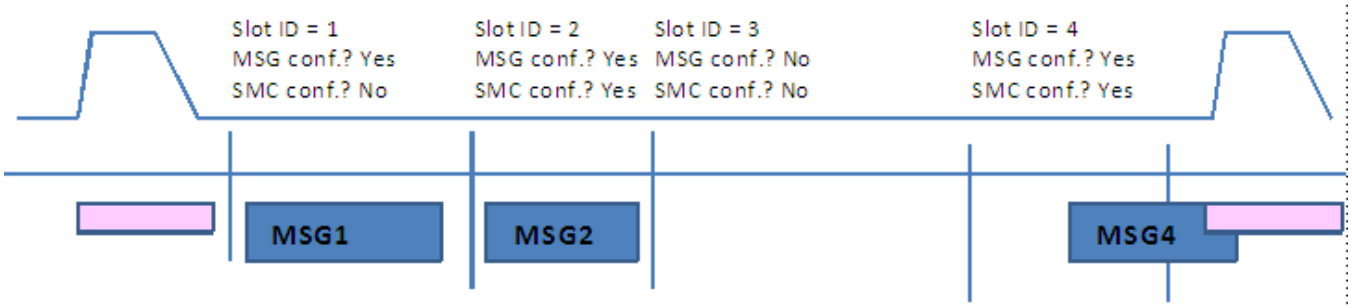
Figure 56-16. Message Much Too Late (in last slot)

In this scenario, the last message is received late.

MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the end of the final slot, but before the Manchester decoder is disabled by the SYNC pulse on the bus. The message is received correctly with the T bit set.

56.4.2.3.4.8 Message Too Late (in last slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	Yes	No	Yes	Yes	4	M0/M1 or 0/0

Figure 56-17. Message Much Too Late (in last slot)

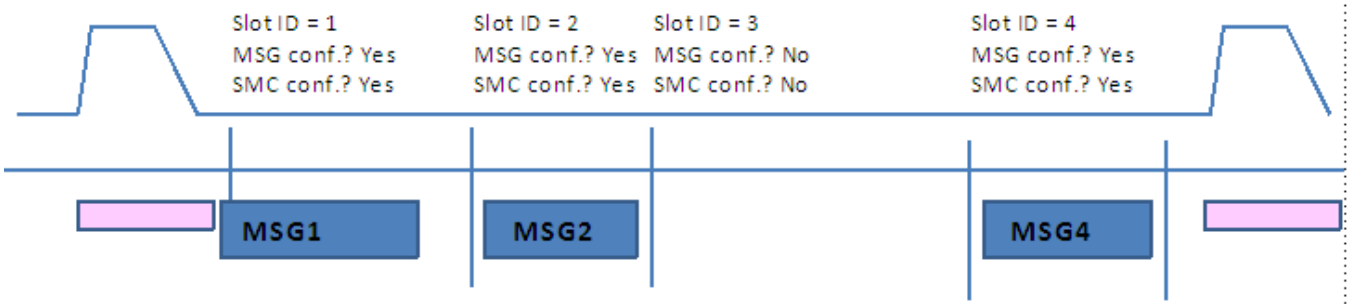
In this scenario, the last message is received late in the last slot.

MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the Manchester decoder is disabled by the SYNC pulse on the bus, so the last part of the message cannot be received correctly. Nevertheless, the message reception is completed and the message is stored in the message buffer with the T bit set and 0 written in the buffer for all bits received while the Manchester decoder is disabled. This leads to E and C bit errors.

In an extreme case, it could also lead to EM bit errors, in which case the SMC buffer will also be written with (0,0) otherwise, the correct bits are written to the SMC buffer.

56.4.2.3.4.9 Message Too Early (in first slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	0	Yes	No	No	No	1	M0/M1
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

Figure 56-18. Message Much Too Early (in first slot)

In this scenario, the first message is received early.

As soon as the Manchester decoder disable phase has ended, the IP verifies the idle condition on the signal line and then looks for the first message start sequence. MSG1 start sequence is received before the start of slot 1 with slot counter 0 captured. MSG1 is then received normally and saved with slot counter 0 and T bit set.

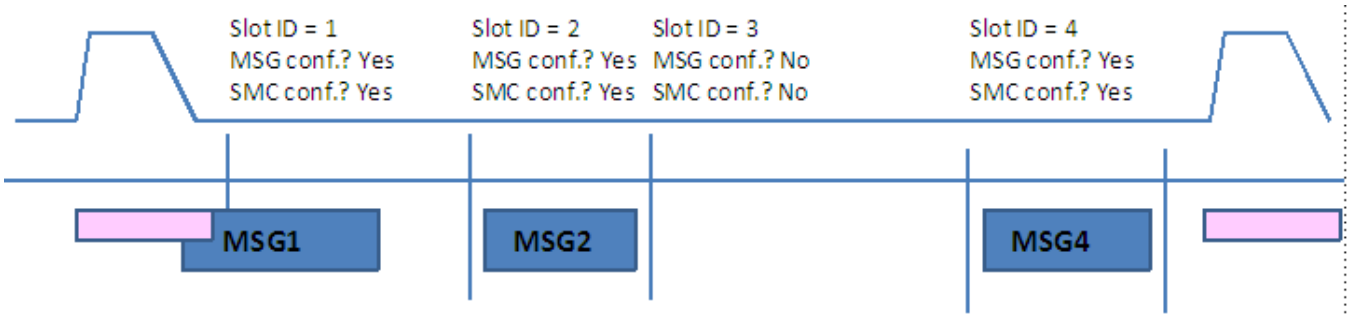
SMC bits M0/M1 are saved to SMC buffer 1.

MSG2 and MSG4 are received normally.

NOTE

This scenario can be avoided if the start of slot 1 is configured to coincide with the end of the Manchester decoder disable phase.

56.4.2.3.4.10 Message Too Early (in first slot and overlaps Manchester Decoder disable)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	Yes	No	No	1	0/0
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

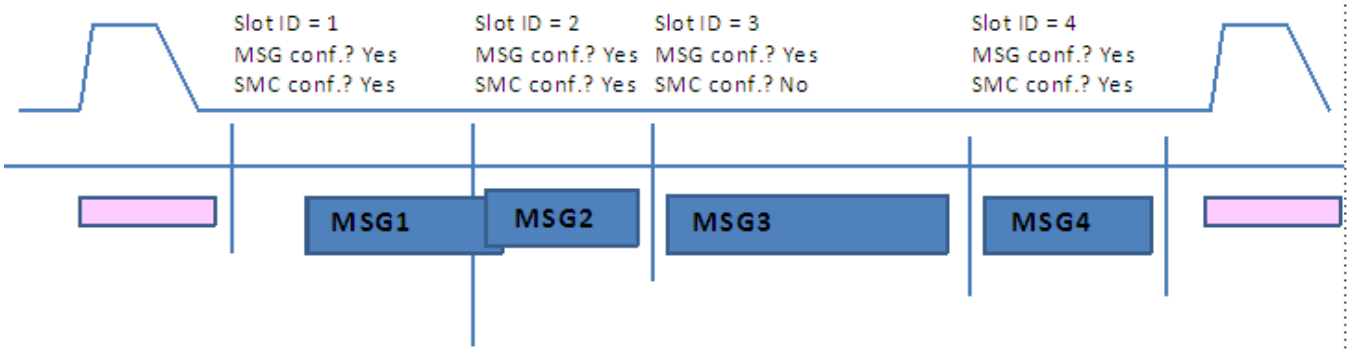
Figure 56-19. Message Too Early (in first slot)

In this scenario, the first message is received early and overlaps with the Manchester decoder disable phase.

As soon as the Manchester decoder disable phase has ended, the IP starts looking for “idle condition” on the signal line by monitoring it for 42 counts of the sample clock. If the signal line is found to be initially high or if a valid transition occurs in the time frame of 42 samples of the sample clock, the Manchester decoder stops decoding until the end of slot 1. Decoding will restart at the beginning of slot 2.

MSG1 is assumed to be lost and the “F” bit is set for that message with the Slot Counter as “1”. If MSG1 SMC is configured then SMC is saved with “00” appended for the M0/M1 bits.

56.4.2.3.4.11 Message Too Late (overlaps next message)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	Yes	No	Yes	Yes	1	M0/M1
MSG2	2	No	Yes	No	No	2	0/0
MSG3	3	No	No	No	No	No	No
MSG4	4	No	No	No	No	4	M0/M1

Figure 56-20. Message Too Late (overlaps next message)

In this scenario, MSG1 is received too late and overlaps with MSG2.

MSG1 start is recognized in slot 1. The last bits of MSG1 are disrupted by the start of MSG2 leading to likely E and C bit errors.

At the end of MSG1, idle time is found to be violated by either the signal line being high as soon as NO_DETECT_STATE is entered or FIRST_TRANSITION_DETECT occurs within 42 counts of the idle detect sample clock. The Manchester decoder stops decoding until the end of slot 2. Decoding is restarted at the start of slot3. MSG2 is not received and MSG2 location in the memory will be saved with the F-bit set. The SMC buffer 2 will be updated with 00.

MSG3 and MSG4 are received correctly in their own slots.

56.4.2.3.5 Data Frame Storage

Once the data frame is received and all required status fields updated (for example, Time stamp, C, E, EM, T, F and Frame Counter), this 64-bit value in the data receive register is moved to the RAM Register (PMB).

- This RAM buffer is a 32 (depth) × 64 (frame size) register array with DMA transfer capability (both single or burst).
- Also, the individual RAM locations can be accessed directly by the microcontroller core any time.
- Care needs to be taken that receive data is assigned to the sensor and frame slot unambiguously.

56.4.2.3.6 Message diagnostics

Apart from the regular status bits (mentioned above), a few more diagnostic checks are done and stored as status bits along with each data frame location in RAM. These are:

- Diagnosis registers are associated with each sensor data RAM buffer to indicate
  - RAM buffer has been filled with data.
  - RAM buffer has been overwritten before data previously stored was read.
  - Data written to buffer includes an error bit (E, EM, C, T, or F, configurable).
- The diagnosis registers are able to be read with a single DMA access.
- After successfully completing a read access, these registers are reset by hardware. This is done by mapping these diagnosis bits byte-wise so that one byte access clears a single bit without affecting other bits.

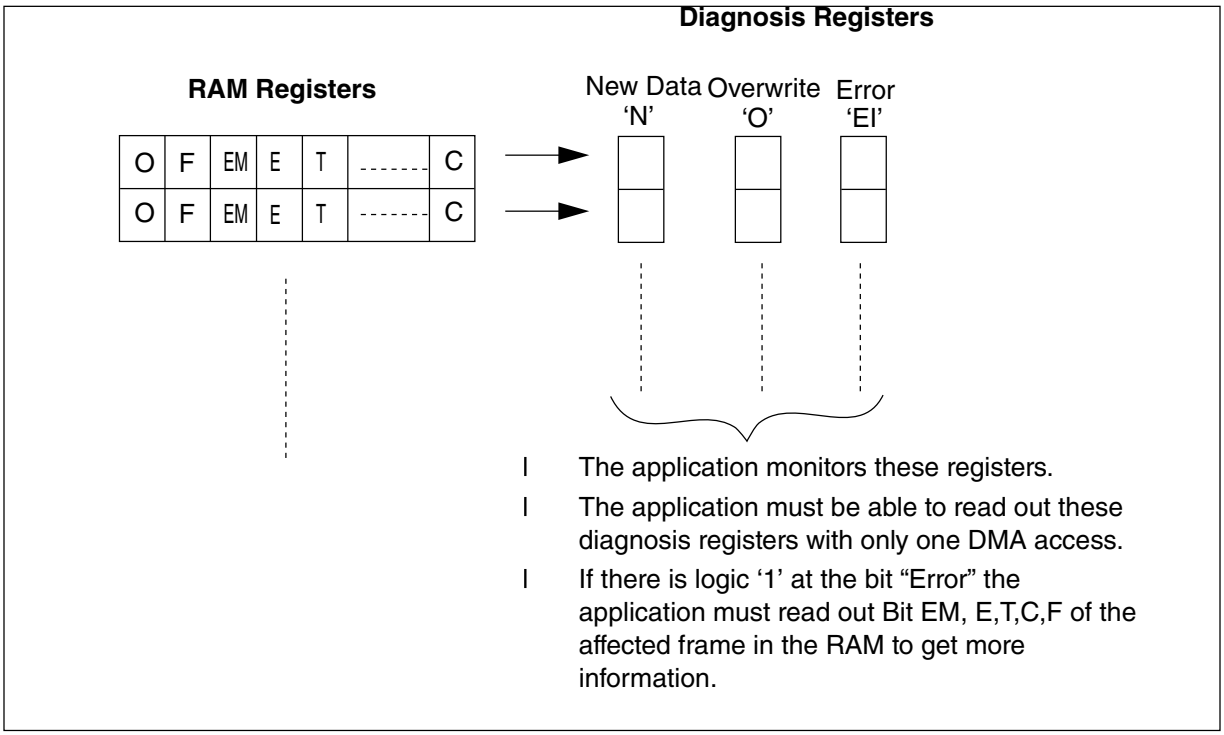


Figure 56-21. Diagnosis Information attached to every frame

Note

See Implementation of EI Diagnostic Register for more detail.

56.4.2.3.7 SMC message extraction

The optional messaging channel M0, M1 (shown in Figure 56-5) is used by sensors to send an SMC frame along with the normal PSI5 frame. It takes 18 PSI5 frames to send one SMC frame.

The extraction of SMC bits is done for each slot separately and stored separately into the SMC frame receive buffers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (depth of SMC frame, as defined below)

The extraction logic works as follows:

- Search for SMC start sequence 111111 followed by 0 on M1 and keep storing last 6 bits {CRC} on M0.
- If the start sequence is detected it continues with further extraction and storage of M1 and M0.
    - Store the extracted frame on the corresponding slot buffer.

- Move the SMC frame to corresponding SFR[i] register period. Once complete, the SMC frame is extracted and generates an interrupt, so that it can be read by the CPU.
- If the start sequence is not detected, it continues with the search.

The SMC frame is extracted in hardware and stored separately as memory-mapped registers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (depth of SMC frame).

56.4.2.4 Message Read Logic

PSI5 supports two kinds of read access to received sensor data.

- Read access through memory-mapped RAM registers. These registers can be read by the CPU any time or, specifically, when the concerned interrupts are raised. Applicable for both the SMC and the PSI5 messages.
- Read access through DMA request. Here the DMA sequencer treats the memory-mapped registers as a block and reads them sequentially. Note that even if the DMA is used, the CPU can read the RAM registers any time by addressing them. This is applicable for both the SMC and the PSI5 messages. When reading the PSI5 messages, it is possible to read only the PSI5 messages or PSI5 + diagnostic registers.

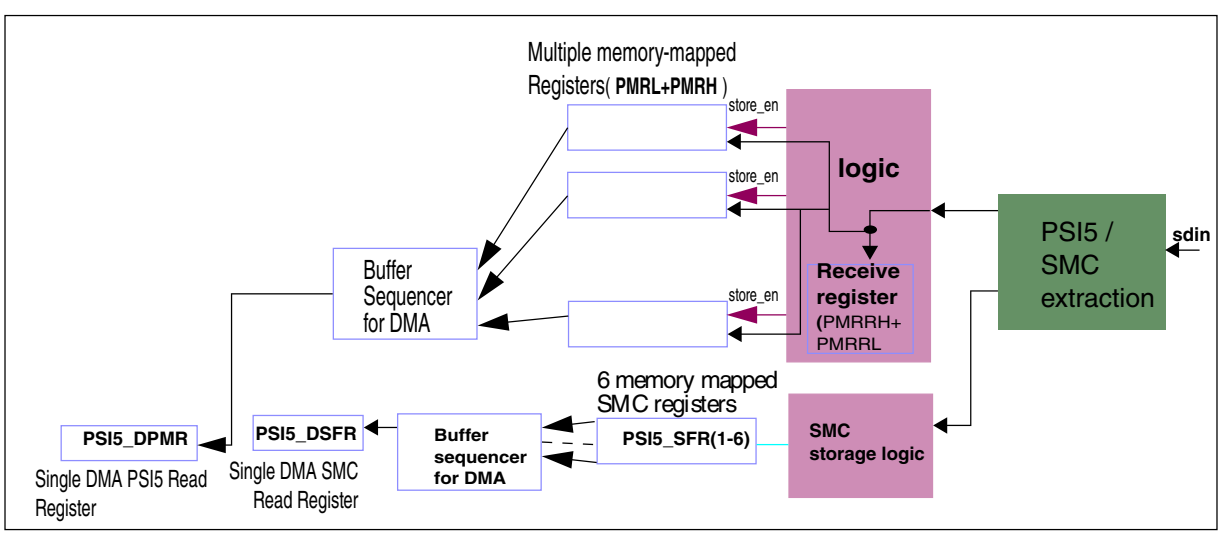


Figure 56-22. PSI5 and SMC Message Read Logic Block

56.4.2.5 CRC Recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

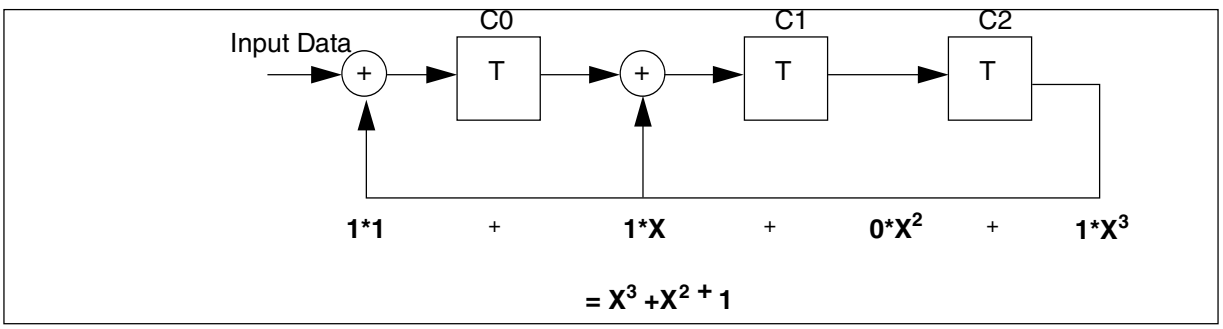


Figure 56-23. CRC calculation scheme

Table 56-19. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 56-20. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 56-21. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 → Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

56.4.3 ECU-to-sensor communication

Whereas the sensor-to-ECU communication is realized by current modulation, voltage modulation on the supply lines is used to transmit commands to the sensors. A predefined start sequence allows the sensors to recognize that a command is being transmitted during all synchronous operation modes.

56.4.3.1 Physical Layer

A logical 1/0 is represented by data sync pulse of variable length, a logical 1 by the sync pulse of width W1(programmable from 0 to 127 µs) and logical '0' by sync pulse with W0{programmable from 0 to 127 µs}.

56.4.3.2 Data Link Layer

56.4.3.3 Bit coding

The bit period is the cycle time as specified for the operation mode, for example, 500 µs in the PSI5-P10P-500/3L mode. (For details about this topology, please refer Section PSI5 topology) The lower limit depends on the filter characteristics of the hardware circuitry used for the sync signal detection (subsequent sync signals might vary the internal $V_{ss,\ base}$ reference).

56.4.3.4 Data transmission

The IP supports the standard and the Non-Standard ECU to Sensor communication Formats . In the Non-Standard mode of communication it can be used to send any frame size programmable from 1-64 bits. In the Standard Formats it supports both the V1.3 and the V2.0 of the PSI5 ECU to Sensor Communication as briefed below in Figure 56-24 and the text that follows it.

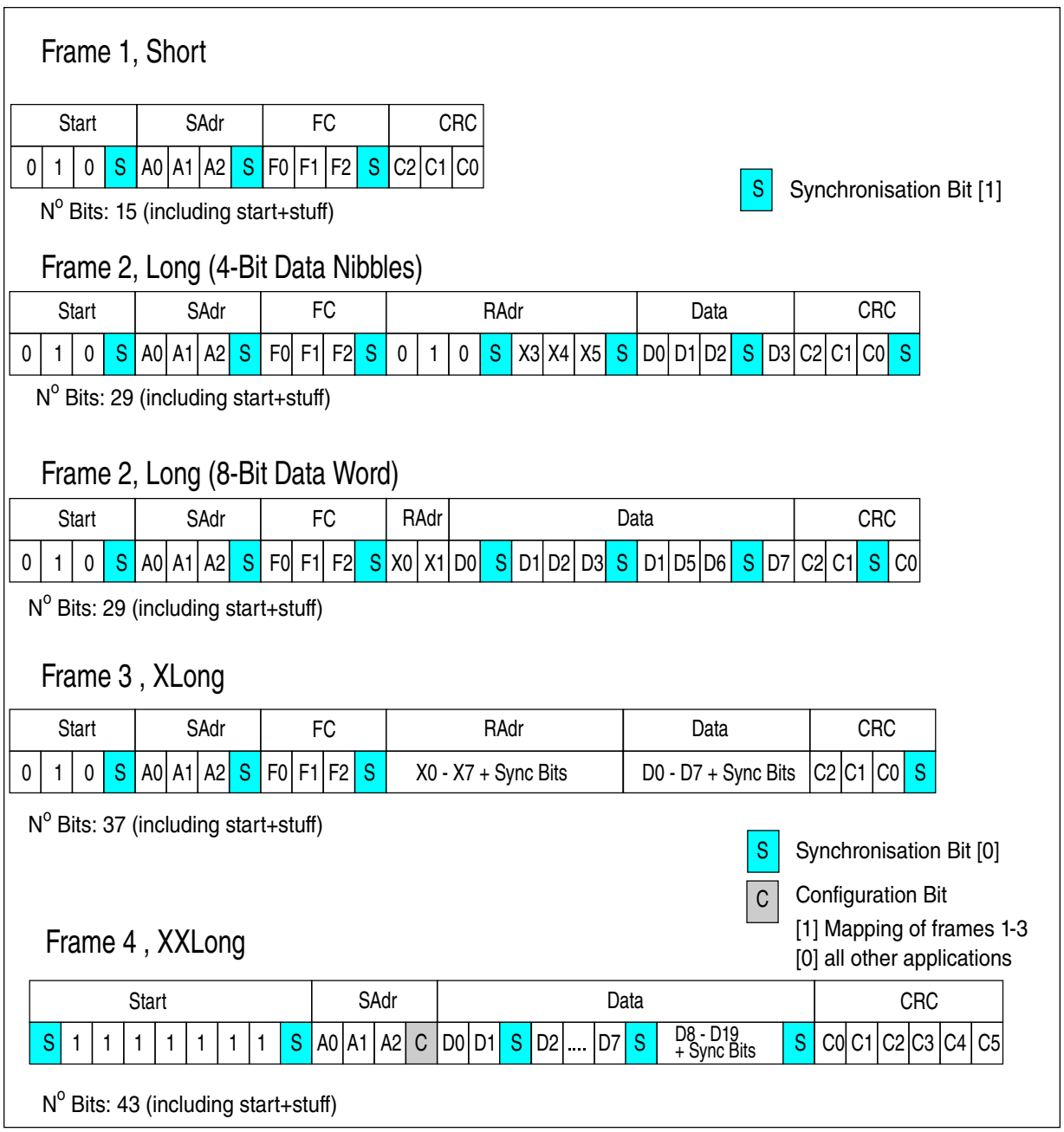


Figure 56-24. Standard Frame Fomats Supported

- **Short(V1.3)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- **"010", S, "A0, A1, A2", S, "F0, F1, F2", S, "C2, C1, C0"** where "010" is the start sequence "S" is the stuff bit(1) ,A0-A2 is the sensor address ,F0-F2 are the functional codes and "C2-C0" is the CRC. Please refer Figure 56-24 for the frame structure.
  - The CRC polynomial used is $x3 + x2 + 1$ with a binary initialisation value of "111(C2:C0)".
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]….DPR[5],000) with DPR[0] being fed first in the CRC calculator.
- **Long(V1.3 - 4bit Data Nibble)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- **"010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, X2", S, "X3, X4, X5, S, "D0, D1, D2", S,"D3, C2, C1", S, "C0"** where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes,"X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Please refer Figure 56-24 for the frame structure.
  - The CRC polynomial used is $x3 + x2 + 1$ with a binary initialisation value of "111(C2:C0)".
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,X0..X5,D0…D3,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]….DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- **Long(V1.3 - 8bit Data Word)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - **"010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, D0", S, "D1, D2, D3, S, "D4, D4, D6", S,"D7, C2, C1", S, "C0"** where "010" is the start sequence "S" is the stuff

bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes,"X0-X1" is the Range Address, "D0-D7" is the data and "C2-C0" is the CRC .Please refer Figure 56-24 for the frame structure.

- The CRC polynomial used is $x3 + x2 + 1$ with a binary initialisation value of "111(C2:C0)".
- The transmitter extends the data bits by three zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,X0..X1,D0…D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
- When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]….DPR[15],000) with DPR[0] being fed first in the CRC calculator.

- **X-Long(V1.3) :** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - **"010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, X2", S, "X3, X4, X5", S, "X6, X7, D0", S, "D1, D2, D3", S, "D4, D5, D6", S, "D7, C2, C1", S, "C0"** where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes,"X0-X5" is the Range Address , "D0-D3" is the data and "C2-C0" is the CRC . Please refer Figure 56-24 for the frame structure.
  - The CRC polynomial used is $x3 + x2 + 1$ with a binary initialisation value of "111(C2:C0)".
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,X0…X7,D0…D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]….DPR[21],000) with DPR[0] being fed first in the CRC calculator.
- **XX-Long(V2.0) :** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - **"011111110", "A0, A1, A2", "C", "D0, D1", S, "D2, D3, D4" D5, D6, D7, S, "D8, D9, D10, D11, D12, D13", S, "D14, D15, D16, D17, D18, D19", S, "C0, C1, C2, C3, C4, C5"**

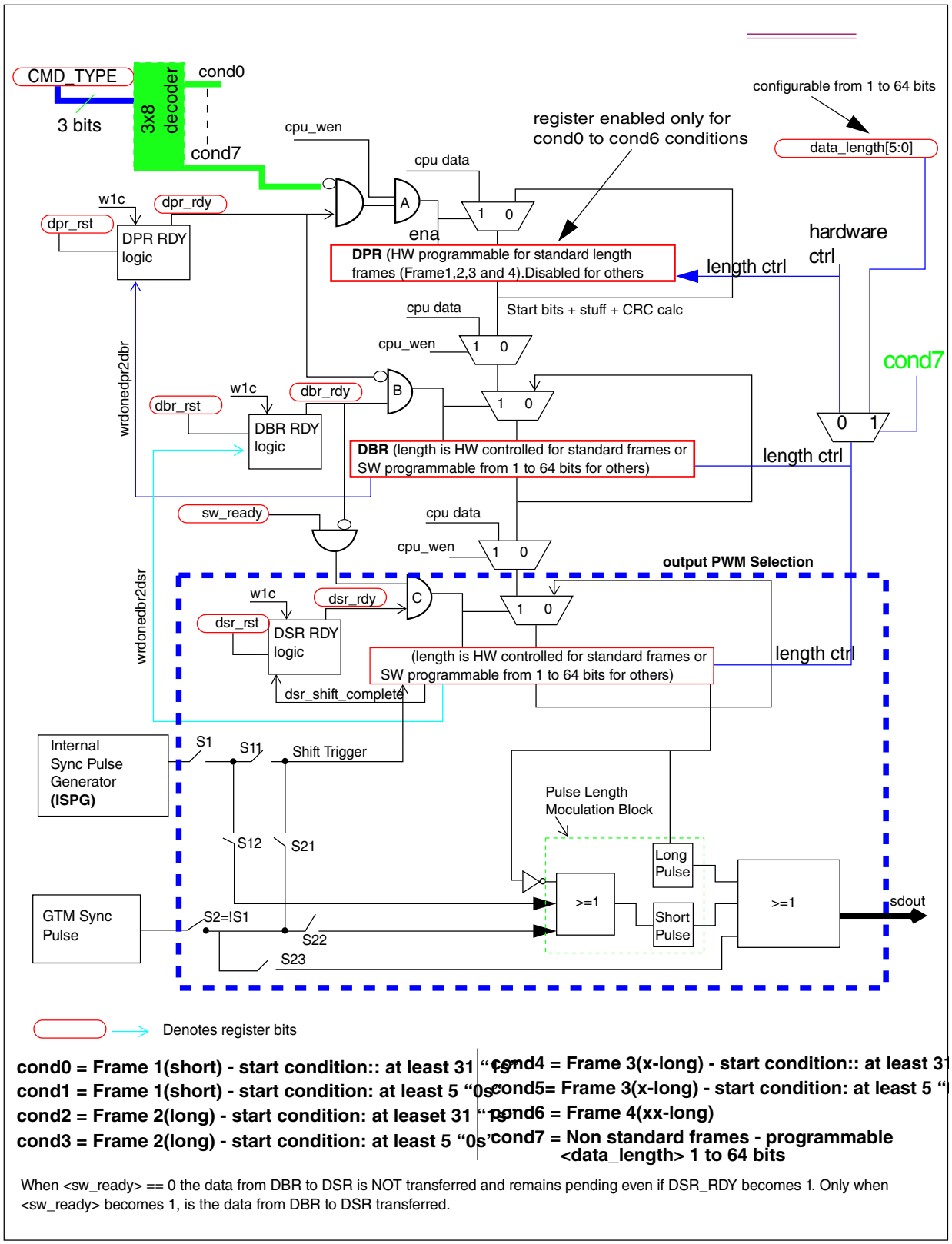
where "011111110" is the start sequence, "S" is the stuff bit(0), A0-A2 is the sensor address, "C" is the configuration bit, "D0-D19" is the data and "C0-C5" is the CRC. Please refer Figure 56-24 for the frame structure.

- The CRC polynomial used is $x6 + x4 + x3 + 1$ with a binary CRC initialization value 010101(C5:C0) .
- The transmitter extends the data bits by six zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,D0..D19,"000000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
- When mapping this sequence to the DPR register bits for calculating the CRC ; then it becomes (DPR[0],DPR[1]….DPR[23],000000) with DPR[0] being fed first in the CRC calculator.

56.4.3.4.1 Transmission Data Framing

The output data transmission block is constructed of seven sub-blocks:

- Data preparation register
- Buffer register
- Data shift register
- Periodic Sync Pulse Trigger Generator
- "GTM driven" sync pulse trigger input (GTM external to PSI5)
- Pulse length modulation block
- OR gate



cond0 = Frame 1(short) - start condition:: at least 31 “1s”
cond1 = Frame 1(short) - start condition: at least 5 “0s”
cond2 = Frame 2(long) - start condition: at leaset 31 “1s”
cond3 = Frame 2(long) - start condition: at least 5 “0s”
cond4 = Frame 3(x-long) - start condition:: at least 31
cond5= Frame 3(x-long) - start condition: at least 5 “
cond6 = Frame 4(xx-long)
cond7 = Non standard frames - programmable <data_length> 1 to 64 bits

When <sw_ready> == 0 the data from DBR to DSR is NOT transferred and remains pending even if DSR_RDY becomes 1. Only when <sw_ready> becomes 1, is the data from DBR to DSR transferred.

Figure 56-25. Data Transmission Block

56.4.3.4.2 Data Preparation Register (64-bit register, DPRH + DPRL, DPRH is fixed at 0)

Upto 24 bits of this register can be written by the CPU. These register bits contain the ECU to sensor command that needs to be transmitted after being appended with the CRC, start bits and stuff bits. The hardware automatically calculates the CRC and adds the start and the stuff bits, before transferring the command to DBR.

- When DPR_RDY == 1, the data to be transmitted to sensors is written to this register via system bus with an order data bit to be transmitted first at LSB and furthest on MSB. The data length to be written is 24 bits in the DPRL. DPRH is always = 0.
- Having written the data to the DPR, the DPR_RDY bit has be cleared with a w1c. This indicates to the internal logic that the data in the DPR is valid and is to be processed.
- If DPR_RDY = 0 and a new command is written to the DPR then the command is rejected and GISR[IS_PROW] is set.
- If the CRC or parity bit calculation is finished, then the data is shifted to the Buffer register with the start bits and the sync bits appended as well. This transfer is done to the Buffer register only if the Buffer register is ready for new data, which is indicated by GISR[DBR_RDY] == 1.
- As mentioned above, once this tranfer is done, the DPR_RDY becomes 1 and a new command can be written to it.

56.4.3.4.3 Data Buffer Register (64-bit register)

- This register has two subregisters DBRH (higher register) and DBRL (lower register). Both of them are collectively referred to as DBR.DBR is of length 64 bits DBR[63:0]
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writeable bits corrresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field please refer bit description in CMD_TYPE[2:0]
- When DPR_RDY == 0 and DBR_RDY goes to 1 then internal logic automatically transfers the data from the DPR (with CRC + stuff bits + sync bits) to the DBR. DBR_RDY goes low and DPR_RDY goes 1.
- If DBR_RDY == 1 AND DPR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping preparation register in case application includes the start and sync bits and calculates the CRC/Parity itself.

If the CPU writes to the DBR when DBR_RDY == 1 but DPR_RDY == 0 and PSI5_DOBCR[CMD_TYPE] != 7, then the results can be unpredictable since the DPR has a pending data that needs to be transferred to DBR.

- If the CPU is writing to the DBR as mentioned above, then the DBR_RDY bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DBR is valid and is to be used for the shift operation.
- If while DBR_RDY == 0, and a new command is directly written to the DBR then the command is rejected and GISR[IS_BROW] bit is set.
- If while DBR_RDY == 0 a new command HAS to be written in the DBR then DOBCR[DBR_RST] has to be written to 1. Writing a 1 to DBR_RST will reject the current command in the DBR and generate DBR_RDY = 1.
- If the Data Shift Register signals that it is ready for new data (PSI5_GISR[DSR_RDY] == 1), the data of the Buffer register is shifted into the data shift register provided PSI5_DOBCR[SW_READY] is 1.
- If PSI5_DOBCR[SW_READY] == 0 then the data from the DBR is NOT transferred to the DSR automatically even if the DSR_RDY goes as 1. In such a case the IP waits for the software to make the PSI5_DOBCR[SW_READY] bit as 1; only then does it transfer the contents to the DSR.

56.4.3.4.4 Data Shift Register (64-bit register)

- This register has two subregisters DSRH (higher register) and DSRL (lower register). Both of them are collectively referred to as DSR.DSR is of length 64 bits DSR[63:0] . This register has a 1-1 correspondence with the bits in the DBR register when this register is updated automatically by the hardware.
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writeable bits corrresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field please refer bit description in CMD_TYPE[2:0].
- When DBR_RDY == 0 and DSR_RDY goes to 1 then internal logic automatically transfers the data from the DBR to the DSR provided PSI5_DOBCR[SW_READY] == 1'b1, else this transfer remains pending till PSI5_DOBCR[SW_READY] is '0'. Once the transfer is complete, DSR_RDY goes low and DBR_RDY goes 1. DSR contents that would finally get transferred, can be read at this stage, i.e., when DBR_RDY becomes “1”.
- Making PSI5_DOBCR[SW_READY] now as '0' will not interrupt the ongoing transfer in the DSR.

- If DPR_RDY == 1 AND DBR_RDY == 1 AND DSR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when DSR_RDY == 1 but DBR_RDY == 0 then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.
- When PSI5_DOBCR[CMD_TYPE] == 7 and If DBR_RDY == 1 AND DSR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when DSR_RDY == 1 but DBR_RDY == 0 then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.
- If the CPU is writing to the DSR as mentioned above then the DSR_RDY bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DSR is valid and is to be used for the shift operation.
- If while DSR_RDY == 0, and a new command is directly written to t he DSR then the command is rejected and GISR[IS_DSROW] is set.
- If while DSR_RDY == 0 a new command HAS to be written in the DSR then DOBCR[DSR_RST] has to be written to 1. Writing a 1 to DSR_RST will reject the current command in the DSR and generate DSR_RDY = 1.
- This register is clocked by the ISPG pulses or the GTM pulses, as described in Sync Pulse Trigger Generation.
- Once the shift operation is complete in the DSR then DSR_RDY becomes 1.

56.4.3.4.5 Sync Pulse Trigger Generation

The sync pulse trigger can be used to generate the timing pulses or the data pulses.

The sync pulse can be generated based on triggers from two different sources:

- **ISPG (Internal Sync Pulse Generator):** Composed of CTPR, CIPR and Channel Target Counter (per channel). The internal sync pulse generator is clocked by either the GTM clock (max frequency is 3.9 MHz) or the internal 1 MHz clock. The Channel Target Counter operates with a 1 µs resolution (when using the 1 MHz clock) or GTM clock resolution (when using the GTM clock). The ISPG is implemented to allow generation of internal periodic trigger. Please Refer Figure 56-30 for the ISPG structure and Figure 56-29 for the CTPR and CIPR explanation. The ISPG uses the Channel Target Counters (CTCs) for generating the sync pulse. The CTCs can be triggered globally (simultaneous for all channels) or locally (individual to each channel). The CTC_GED bit in the Global Control Register (GCR) ensures that all the CTCs start/stop simultaneously. The individual target

values when the trigger pulse is generated are set in the Counter Target Pulse Register (CTPR) and the Channel Initial Pulse Register (CIPR). These registers are present for each channel. When staggering the channels, the CTC_GED bit should be used for providing a simultaneous trigger to the staggered channels. It is also possible for a subset of channels to work on the above arrangement while the remaining channels independently generate trigger pulses for which start/stop is controlled locally to the channels.

- For each channel, it is possible to configure the trigger period by software using sync pulse generator control/configuration register and setting compare values in Counter Target Pulse Register(s). The Counter Initial Pulse Register (CIPR) can be used to provide the initial offset between various channels.
- It is possible to deactivate each channel separately for the purpose of Sync Pulse generation.
- The periodic pulses can be used to synchronize sensor-to-ECU communication (synchronization timing pulses) [State 1 of Valid states for integrated sync pulse generator]or can be used as data pulse during ECU-to-sensor communication (through the DOBCR) [State 2 of Valid states for integrated sync pulse generator].
- When programming the CTPR, the Tsync period should be programmed >= 6 SP_TS_CLK cycles (For details of SP_TS_CLK, refer to Bit 23 of Section 54.3.2.2, PSI5 Channel Control Register (PSI5_PCCR))

- **GTM event controlled:** Generation of sync Trigger/Pulse can be done based on events (e.g., crank angle or software) through **general timer module (GTM)** output to the PSI5 interface.
  - Here the sync trigger pulses come from external GTM module routed through a full XBAR programmable through SIUL[GTM to PSI5 Connections (in SIUL)]
  - The sync pulse programming (period, etc.) is done inside GTM.
  - In Figure 56-25, if switches S2 and S21 are closed and S22 and S23 are open, the GTM trigger is used for data pulse generation (data pulses for ECU-to-sensor communication). [State 4 of Valid states for integrated sync pulse generator]

- In Figure 56-25, if switches S2 and S22 are closed and S21 and S23 are open, the GTM trigger is used for timing pulse generation (synchronization pulse for sensor-to-ECU communication). [State 3 of Valid states for integrated sync pulse generator]
- In Figure 56-25, if switches S2 and S23 are closed and S21 and S22 are open, the GTM trigger is passed directly to data sync port without pulse length modulation. In this case the sync pulse length is defined by the GTM pulse length. [State 5 of Valid states for integrated sync pulse generator]

- **Programming and control :** For achieving the various states of ISPG control or the GTM control, the switches S1, S2, S11, S12, S21, S22, and S23, shown in Figure 56-25 can be set (closed) or cleared (open) by writing to the three equivalent bits (GTM_TRIG_SEL, SPULSE_SEL and OP_SEL) belonging to the DOBCR, as shown inFigure 56-26. The programming of various states mentioned above, with reference to the Valid states for integrated sync pulse generator can be seen in Table 56-2 and also shown in Figure 56-32. The sending of the ECU to sensor commands, can be controlled by PSI5_GISR[DPR_RDY, DBR_RDY, DSR_RDY] bits and the PSI5_DOBCR register bits.

56.4.3.4.5.1 Timing Pulses

Timing pulses can also be produced without pulse width modulation also . These timing pulses can be generated through two sources:

- Periodic sync pulse from ISPG . In this case the synchro pulse from the internal sync pulse generator can be made available without any modulation. Referring Figure 56-26, in synchronous mode , if DOBCR[GTM_TRIG_SEL] == 0 and DOBCR[OP_SEL] == 0 then the internal sync pulse generator output can be made available without any modulation . The high time of this pulse is fixed at 4 µs.
- Referring Figure 56-26, in synchronous mode , if DOBCR[GTM_TRIG_SEL] == 1 and DOBCR[OP_SEL] == 0 , then the GTM trigger can be used for timing pulse generation without any Pulse Width Modulation.

56.4.3.4.5.2 Data pulse generation

Data pulses are produced through Pulse Length modulation block based on triggers from one of the two selectable sources **ISPG** or the **GTM**, during ECU-to-sensor communication. The selection between the above two sync sources can be done by software programming Pulse length control/configuration register.

- A logical '1' in Data Output Register LSB is coded as data sync pulse of width W1 {up to 127 µs}.
- A logical '0' in Data Output Register LSB is coded as data sync pulse of width W0 {up to 127 µs}.
- The two widths are programmable in Pulse Length Control/Config Register.
- The Data Shift Register is shifted Right by everytime as bit is transmitted.

Figure 56-25 shows the data transmission block.

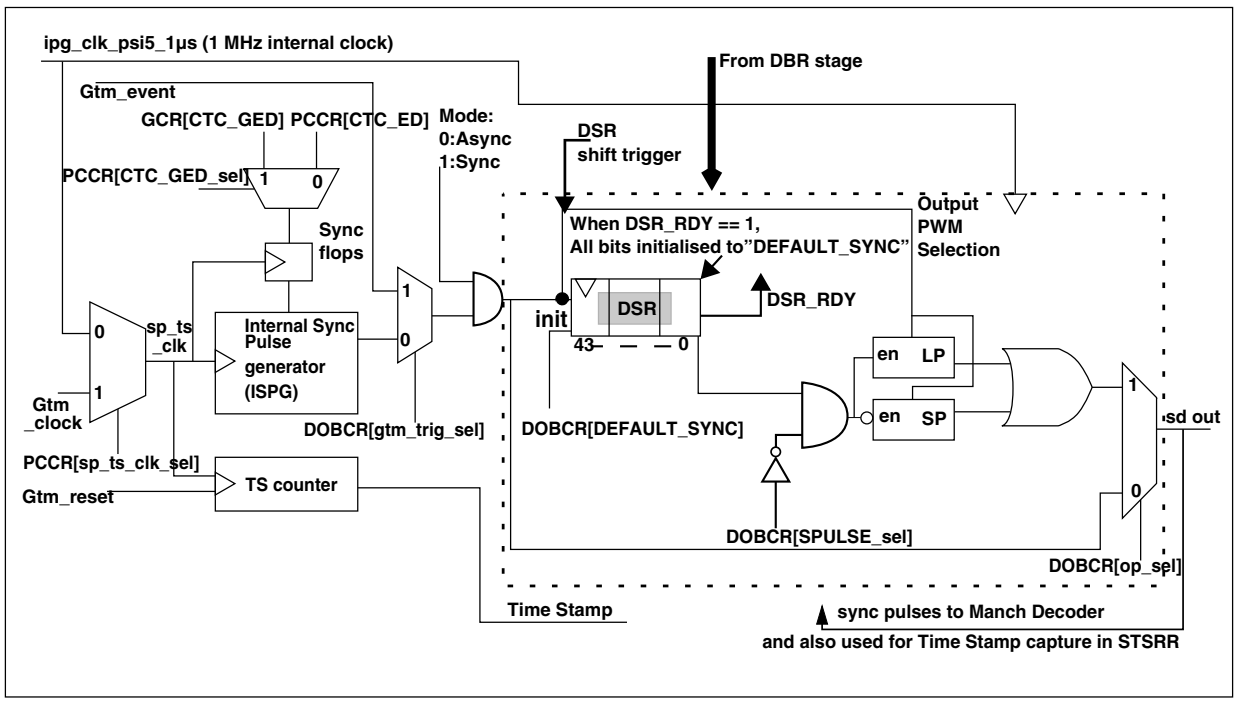


Figure 56-26. ECU-to-sensor communication registers (implementation of the Data Transmission Block figure)

The above figure details the implementation of Figure 56-25 and shows how the various bits mentioned in Section 54.3.2.30, Data Output Block Configuration Register (PSI5_DOBCR) control the behavior of the Output Command path. The above figure also shows the various clock sources to the Internal Sync Pulse Generator and the Time Stamp counter.

56.4.3.5 Response to ECU-to-sensor frames

The response to an ECU-to-sensor frame can be optionally transmitted in two ways:

- Out-of range values of data frame (as fast response coded as normal PSI5 payload region)
- Serial messaging channel (verification of the response frames must be handled by the software)

56.4.4 Tx/Rx scenario

The diagram below shows the communication scenario in synchronous modes. For asynchronous mode, sync pulses are not generated and sensor transmits data asynchronously.

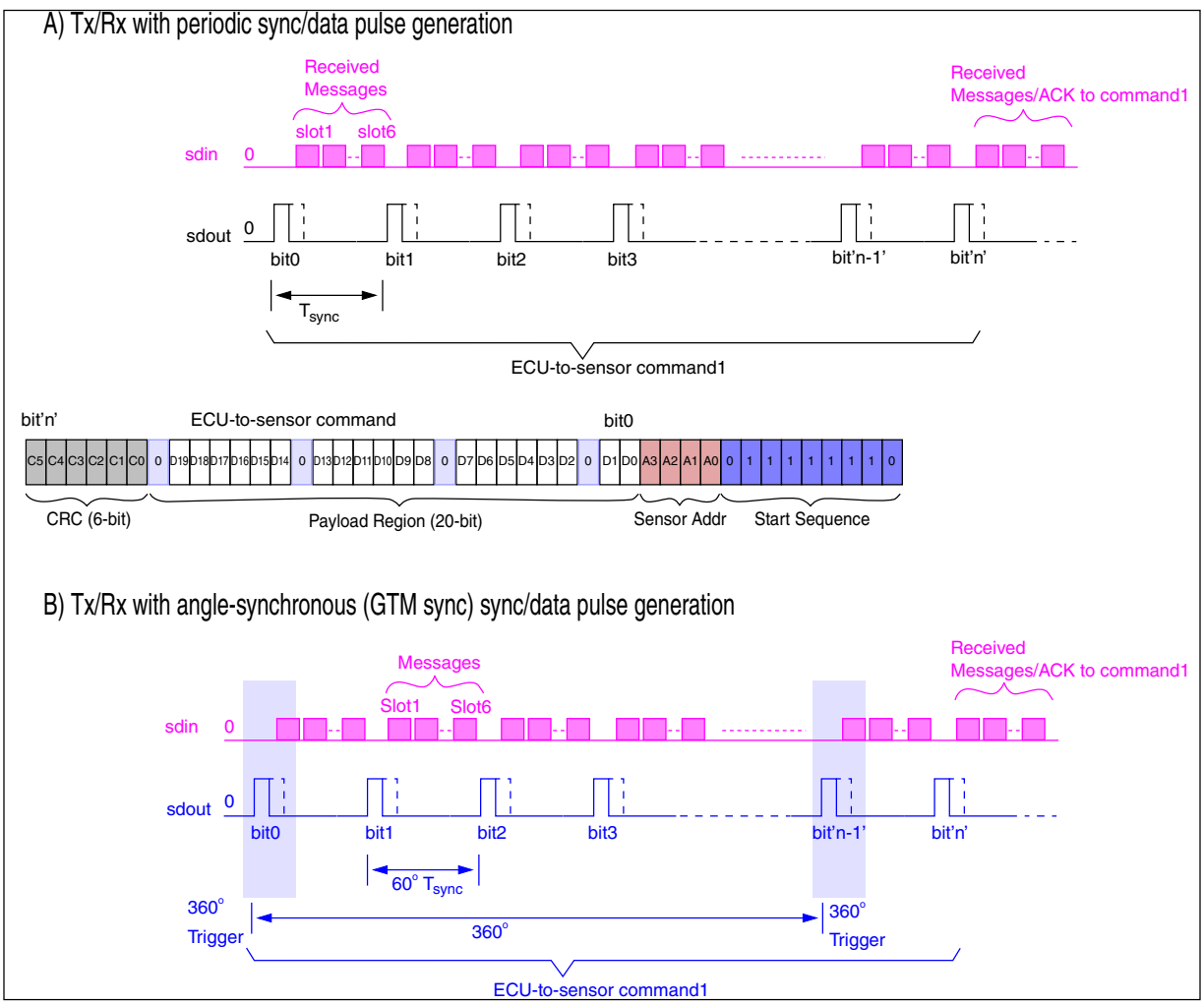


Figure 56-27. Synchronous mode communication scenarios

56.4.5 Interrupt handling

Configurable interrupts based on events (for example, reception of valid message), errors, DMA transfers, and so on are implemented. The following types of events can generate an interrupt:

- Reception of a new PSI5 message
- Reception of a new SMC message
- Unread SMC message overwritten in the SMC buffer area (PSI5_SFR)
- Unread PSI5 message overwritten in the PSI5 buffer area (PSI5_PMRH, PSI5_PMRL)
- Dedicated interrupts for selectable errors, mentioned below:
  - Bit C is set
  - Bit E and/or EM is set
  - Bit T is set
  - No PSI5 frame was received in a defined frame slot (Bit F is set)
- ECU-to-sensor communication-related interrupts
  - The application tries to load new data into the data preparation register before the register has signalized that it is ready for new data. (Note that the new command will be ignored in this case.)
  - The application loads new data into the data buffer register before the register has signalized that it is ready for new data. (Note that the new command will be ignored in this case.)
  - The application loads new data into the data shift register before the register has signalized that it is ready for new data. (Note that the new command will be ignored in this case.)
- DMA related interrupts (transfer finished/FIFO_FULL/underflow)
- Apart from this, a single one sum interrupt OR-ing all the above events.

Various interrupt signals that represent the above events are:

- PSI5 new message arrival interrupt **(Status flags in PSI5_NDSR)** This is asserted when any of the programmed storage PSI5 locations (max 32) receive a new PSI5 message (error/error free). This interrupt also indicates that the PSI5 receive register (PMRRL/PMRRH) has been updated with a new PSI5 message.

- SMC message, ECU-to-sensor communication, and Time Stamp related interrupts **(All Status flags in PSI5_GISR)** : This is asserted during following events:
  - When a new SMC message is received in its corresponding slot OR
  - When there is a CRC error in any SMC message OR
  - When an unread message in some corresponding SMC location (in PSI5_SFR) gets overwritten by a new SMC message OR
  - When DPR is ready for receiving a new data OR
  - When DBR is ready for receiving a new data OR
  - When DSR is ready for receiving a new data OR
  - When there is an attempt to write to DPR while it is NOT ready OR
  - When there is an attempt to write to DBR while it is NOT ready OR
  - When there is an attempt to write to DSR while it is NOT ready OR
  - When data time stamp register is updated with new time stamp value OR
  - When the sync time stamp read register is updated with a new time stamp.
- PSI5 Message Overwrite Interrupt **(Status Flags in PSI5_OWSR)** : This is asserted when any of the programmed storage locations in the memory (max 32) have an unread PSI5 message that is overwritten by a new upcoming PSI5 message. Corresponds to OWSR.
- PSI5 Error interrupt **(Status Flags in PSI5_EISR)** : This is asserted when any of the programmed storage locations in the memory (max32) have an error in the PSI5 message. It is used only for PSI5 messages.
- DMA Interrupt **(Status Flags in PSI5_DSR)** :
  - When the DMA transfer corresponding to conf2, conf3 or conf4 finishes OR
  - When the DMA transfer corresponding to SMC message gets over OR
  - When there is an FIFO_FULL/Underflow in the PSI5 RAM register area OR
  - When there is an underflow in the SMC message area.
- Global Interrupt: Single interrupt ORing all above interrupt events.

56.4.5.1 Error conditions

- C: CRC/parity error on received data, evaluated after recalculating CRC on received data and comparing with received CRC or recalculating parity on received data and comparing with received parity.
- E: Electrical error during any Tbit period (excluding Messaging field M0M1) during frame reception (not applicable during Idle state); that is, if at least one bit is absent during its Tbit period.
- EM: Electrical error on M0M1 field.
- T: Timing error during frame reception violates the slot boundary. Here, all the corner cases and conditions are taken acre of (for example, the isolation of correct slot to correct message if the message is too early or too late with respect to its expected position, the crossing of the slot boundary when one frame overlaps the other, and so on).
- F: No frame received during a particular slot time in case of sync mode.

56.4.6 DMA support

NOTE

DMA master ID replication mode is not allowed together with PSI5.

- The PSI5 module generates separate DMA transfer requests for PSI5 message/ diagnostic status and SMC messages.
- A single DMA request is used for PSI5 messages and diagnostic data. The request is triggered based on the selected DMA configuration (either PSI5 message or Diagnostic data or both). After choosing the configuration, data is available on respective DMA pop-up registers.
- DMA requests for PSI5 messages and diagnostic registers are based on the watermark level of message buffers (to be treated as FIFO for the DMA). PSI5 message buffer FIFOs get data from the PSI5 receiver in the order they are received.

- DMA requests for SMC messages are based on successful storage of each SMC frame to the message buffers. The SMC message buffers get data from SMC extraction logic in fixed round-robin arbitration sequence.
- Please see Single DMA request for PSI5/diagnostic message buffers for PSI5 DMA handling and Single DMA request for SMC received messages for SMC DMA handling

56.4.6.1 Single DMA request for PSI5/diagnostic message buffers

A common DMA request (per PSI5 block) is generated for PSI5 message/diagnostic buffers. The depth (up to 32 in present context) of the message buffer is programmable via the MEM_DEPTH filled in the PCCR. The width of each location of the PSI5 message buffer is 64 bits. When reading PSI5 message through the DMA, the 32-bit DPMR is sequenced with the data from the PSI5 message buffer. When reading the diagnostic registers through the DMA, the 32-bit DDSR is sequenced as NDSR followed by EISR. The DMA operation will happen as follows:

- Based on the DMA configuration chosen, one of the three DMA transfer sequence is initiated:
  - PSI5 buffer followed by diagnostic data sequenced to DPMR
  - PSI5 data sequenced to DPMR
  - Diagnostic buffer data sequenced to DDSR
- The DMA request is asserted when the number of messages stored in the buffer crosses the configured watermark number of messages.
- DMA request is deasserted once the "watermark" number of messages have been read.
- If the number of unread messages is still greater than the watermark programmed, then the DMA request is reasserted.
- The received message data is read from a common memory-mapped 32-bit DMA PSI5 message register (DPMR). Thus, a single PSI5 message (64 bits wide) will require two IPS read operations. The diagnostic data is read either through the DPMR or through the DDSR (single 32-bit register) by the DMA controller via the IPS interface.
- It is assumed that the DMA Controller's Transfer Control Descriptor (TCD) would be set to transfer bytes equal to the messages indicated by watermark level, as described in DMA_PM_DS _WM[4:0] in DMA Control Register (PSI5_DCR). It is assumed

that when responding to a DMA request, the DMA controller will read faster than the rate at which messages are received and stored in the buffer/FIFO which will cause the DMA request to deassert.

- The next received message will be updated on the common memory map register once the previous message is read by the DMA (by the DMA sequencer).
- The store signal from the synchronization logic will push the message into the memory-mapped RAM registers and in parallel also into the PSI5 receive register. Figure 56-22 shows the format in which the messages will be stored in any message buffer (RAM registers) and finally when being read from DMA access register.
- For more details about DMA programming, please refer to PSI5 Channel Control Register (PSI5_PCCR), DMA Control Register (PSI5_DCR), DMA Status Register (PSI5_DSR), DMA PSI5 Message Register (PSI5_DPMR) and DMA Diagnostic Status Register (PSI5_DDSR).

56.4.6.2 Single DMA request for SMC received messages

A separate DMA request (each channel) is generated for the SMC message buffer. Each SMC received messages would be stored in the internal registers and read out by DMA controller via a common memory-mapped DMA SMC Frame Register (DSFR) (single 32-bit register for each message buffer). The data flow will happen as follows:

- A request is asserted when the system buffer successfully receives one complete SMC message, and remains asserted until all complete messages have been read. Data is read from the DMA PSI5 Message Register in the memory map.
- It is the job of an internal sequencing logic to provide the next complete received message. The internal sequencer works in a round-robin fashion checking for the slot-specific registers for each SMC message. Thus the sequence will be slot1--slot2-- slot3-- slot4--slot5--slot6--slot1 and so on. In whichever slot it encounters the first complete SMC message the DMA request would be asserted and the sequencer will wait for this data to be read before moving on to the next slot in its loop. If a particular slot message remains unread and a new SMC message for the same slot comes in then the previous message would be overwritten and the OW bit would be set.

- The DMA controller should be programmed to read only one message in one DMA transfer. Otherwise, if it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur. User software should ensure there is no overflow.
- For more details about DMA programming please refer "PCCR-PSI5 Channel Control Register" on page 267,"DCR-DMA Control Register" on page 272,"DSR-DMA Status Register" on page 276 and "DSFR-DMA SMC Frame Register" on page 284

56.4.7 Message read through interrupt generation

There is a separate interrupt request for received messages for each PSI5 channel. Each PSI5 channel can be configured individually to generate an interrupt on reception of messages. An interrupt is asserted when the corresponding PSI5 channel has received a complete message (PSI5 or SMC). Please see Interrupt handling for more details about interrupts. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing the corresponding PSI5 channel which is enabled to give out messages via interrupts through the IPS Interface. In response to these interrupts, the CPU is expected to read interrupt status register for individual PSI5 Channel (depending on which interrupt was asserted) and store it in system memory.

56.5 Initialization information

56.5.1 Initialization

56.5.1.1 Sensor startup and initialization

After each power-on or under-voltage reset, the sensor performs an internal initialization which is divided into three phases:

- Initialization Phase1: No data transmission.

- Initialization Phase 2: Transmission of type code and serial number of sensor.
- Initialization Phase 3: Transmission of "Sensor Ready" or "Sensor Defect" or other sensor specific data.

As this PSI5 implementation is targeted towards powertrain devices/sensors the standard initialization phase is modified and tuned for those applications. The comparison of the original scheme and enhanced schemes are shown in the figure below. The following figure also describes that the initialization content is the same as defined in V1.3, and the only modification is that it is transmitted using the SMC channel when a fast startup is required. "Fast Initialization Phase" below means that the sending of first sensor data is fast, but the sending of sensor initialization data will actually be much slower.

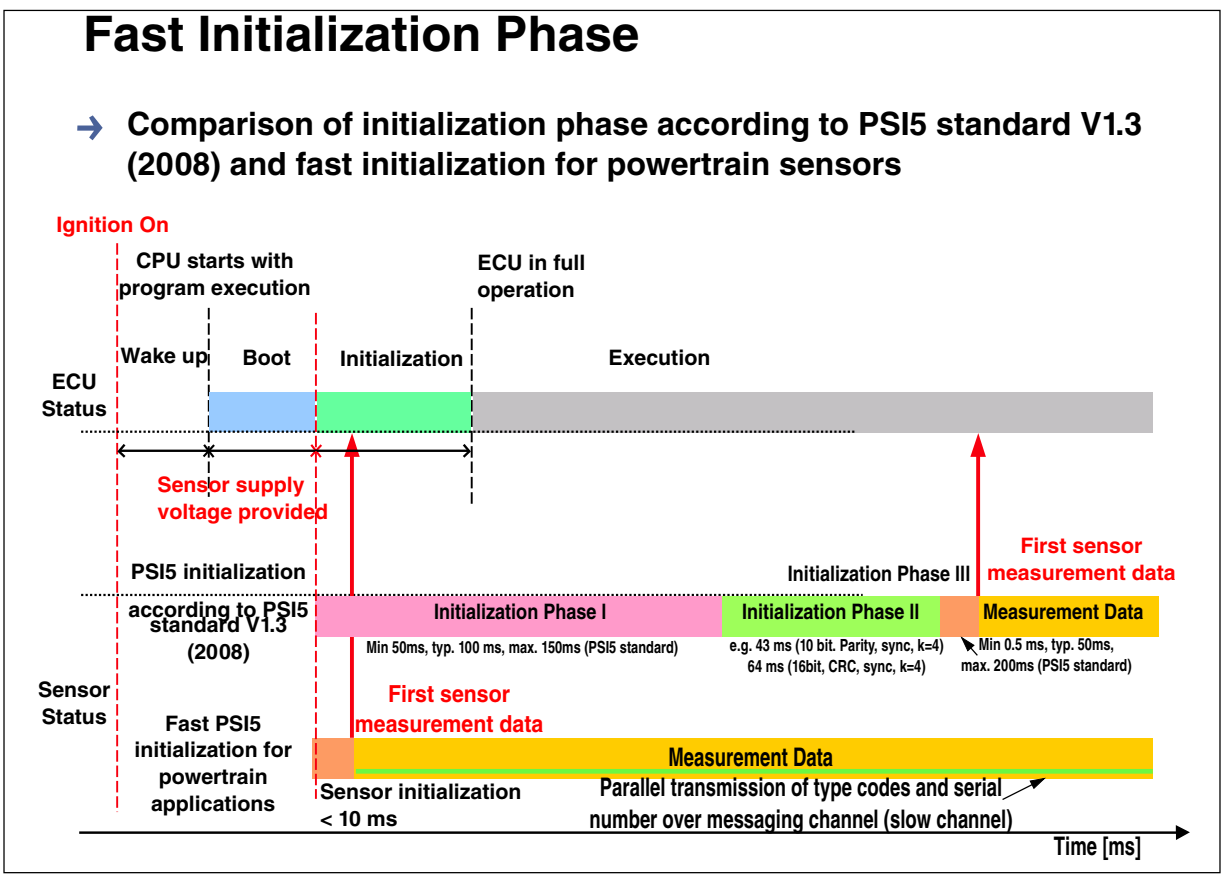


Figure 56-28. Fast initialization phase

56.6 Design Assumptions and Limitations

56.6.1 Assumptions

- Slot Configuration Register would always be programmed with valid number of bits for frame configuration.
- There will be no Timing error bits (T) in async mode. Only CRC Error (C) and Electrical Error (E) bits would be appended to message. There is no time out condition for the Manchestor decoder in case of Async mode. 'NFR' bit is not set in Async mode.

56.6.2 Limitations

As this is a PSI5 receiver, designed primarily to only receive the PSI5/SMC data and store as it is into the RAM buffer. So, it would not be responsible neither to recognize the content of the received messages nor do any decision making based on the content. Similarly, for Command it would not recognize the content of command and would transmit as it is writing into corresponding DSR and reception of PSI5 messages would not in any way depend on the content of command as far as PSI receiver is concerned. Its the responsibility of SW to sequence the issue of commands across the PSI5 channels and thereby recognize the content of received message and infer it.

56.7 Miscellaneous descriptions

56.7.1 Internal SYNC Pulse generation and coordination across PSI5 channels

Each PSI5 channel has the Channel Target Pulse Registers (CTPR) and Channel Initialize Pulse Register (CIPR). The CTPR defines the period of the internal sync pulse generation and the CIPR defines the initial offset time with respect to the sync pulse trigger. The sync pulse trigger can be generated from either the PSI5_GCR[CTC_GED] global bit or the PSI5_PCCR[CTC_ED] channel specific bit.

When staggering of the channels is required then the PSI5_GCR[CTC_GED] global bit is used as a common trigger for all the PSI5 channels in which PSI5_PCCR[CTC_GED_SEL] bit is set to 1. Setting this bit to 1 triggers the internal sync pulse generators of all the PSI5 channels and the trigger of each channel is then automatically staggered with respect to the other channel, depending on the value of the CIPR.

When the specific PSI5 channel is not to be kept as a part of staggered channels, then the channel-specific PSI5_PCCR[CTC_GED_SEL] bit has to be kept as 0. The channel in this case, has to be triggered locally by writing a 1 to the PSI5_PCCR[CTC_ED] of that specific channel. This triggers the internal Sync Pulse Generator of that specific channel without affecting other channels.

Following is the procedure for starting the internal Sync Pulse Generators:

- If in Normal mode, then stop and reset all Channel Trigger Counters (CTCs) by setting the global PSI5_GCR[CTC_GED] bit to 0 or the channel specific PSI5_PCCR[CTC_ED] to 0.
- Enter Config mode.
- In Config mode, load all CTCs with the initialization value (offset) from CIPR.
- In Config mode, load all Counter Target Pulse Registers (CTPR) with the needed sync pulse period.
- Enter Normal mode.
- Send global start trigger (using PSI5_GCR[CTC_GED] bit) to all CTCs for synchronized start for the staggered channels OR send a local start trigger (by using the PSI5_PCCR[CTC_ED]) for channels not staggered.
- If the Compare Unit detects that the value in the CTC is equal to the value in CTPR, the CTC is automatically reset by hardware and restarts with 0.
- In Normal mode, if required, the CTC can be stopped and reset by writing a "0" to the PSI5_GCR[CTC_GED] or the PSI5_PCCR[CTC_ED] as the case may be.
- A sync pulse trigger is created. (See below.)

For the details about the operation of the below interface, please see Section 54.3.2.34, Counter Target Pulse Register (PSI5_CTPR), Section 54.3.2.35, Counter Initialize Pulse Register (PSI5_CIPR) and Sync Pulse Trigger Generation.

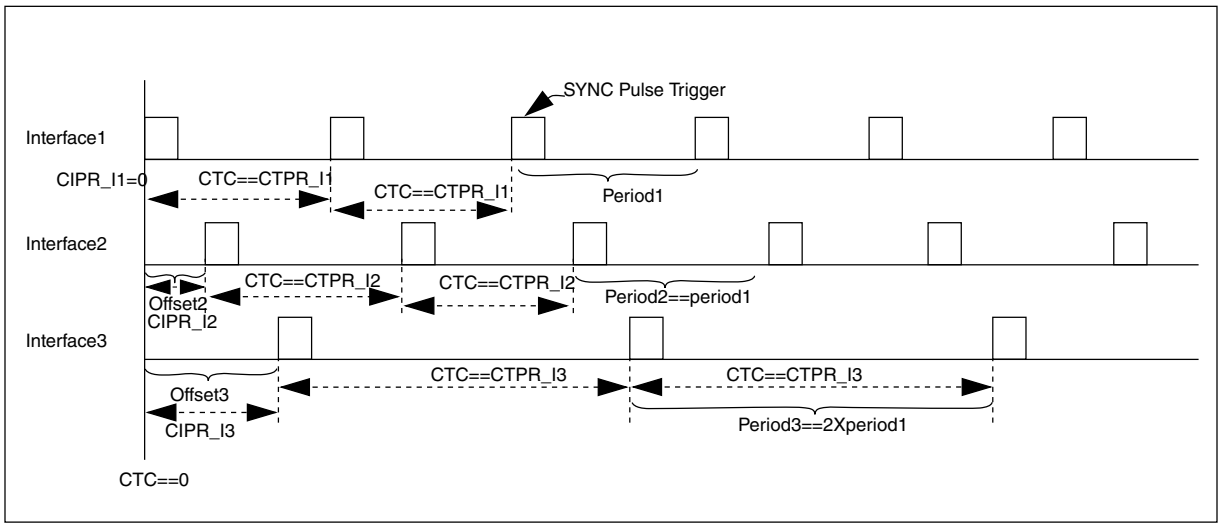


Figure 56-29. CIPR and CTPR

Figure 56-30 shows the various parameters (CIPR/CTPR, and so on) and the associated jitter, across different channels when these channels are staggered. (The figure also shows some other parameters, such as the modulated pulse width parameters and the Manchester decoder disable parameters.)

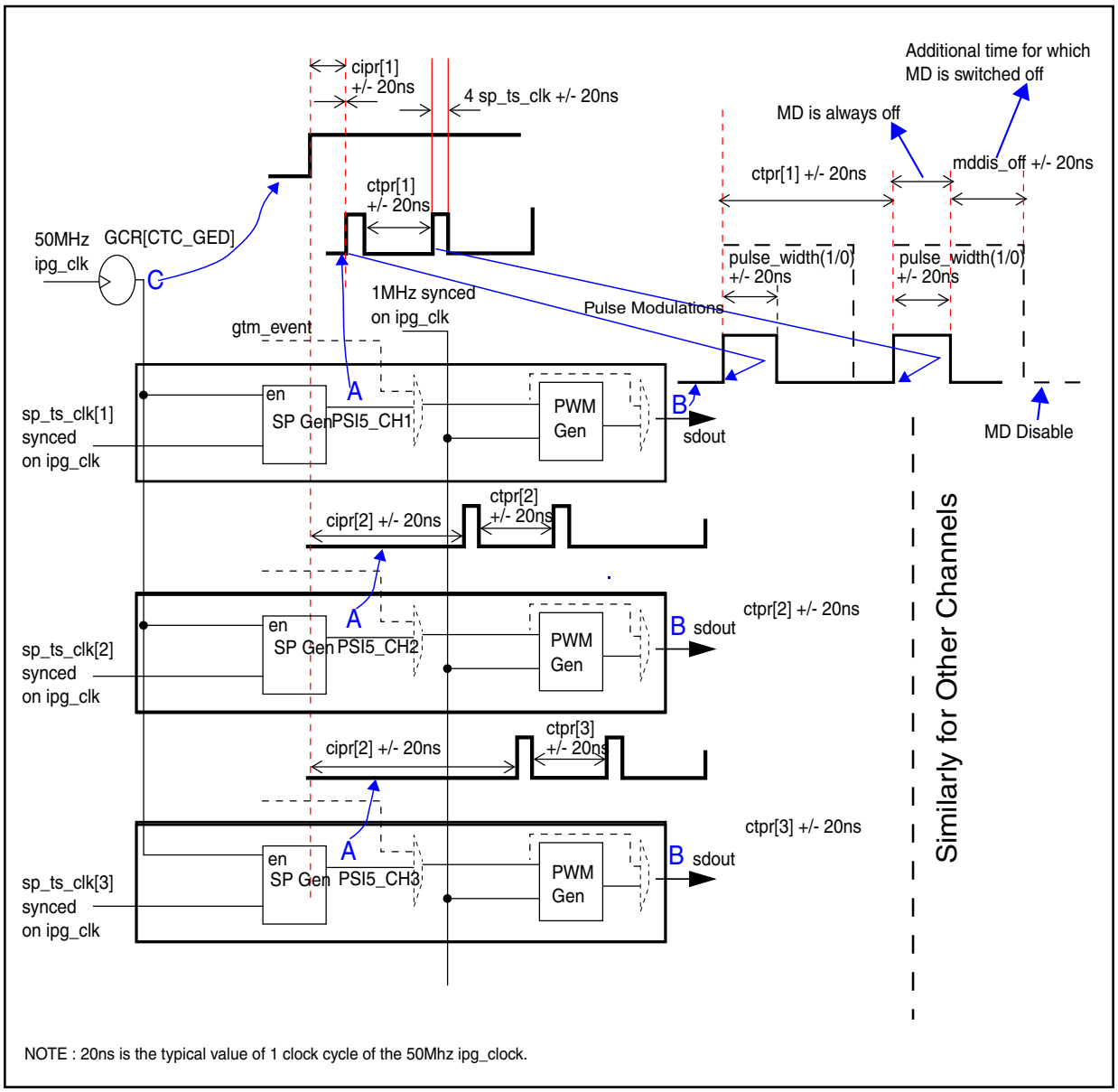
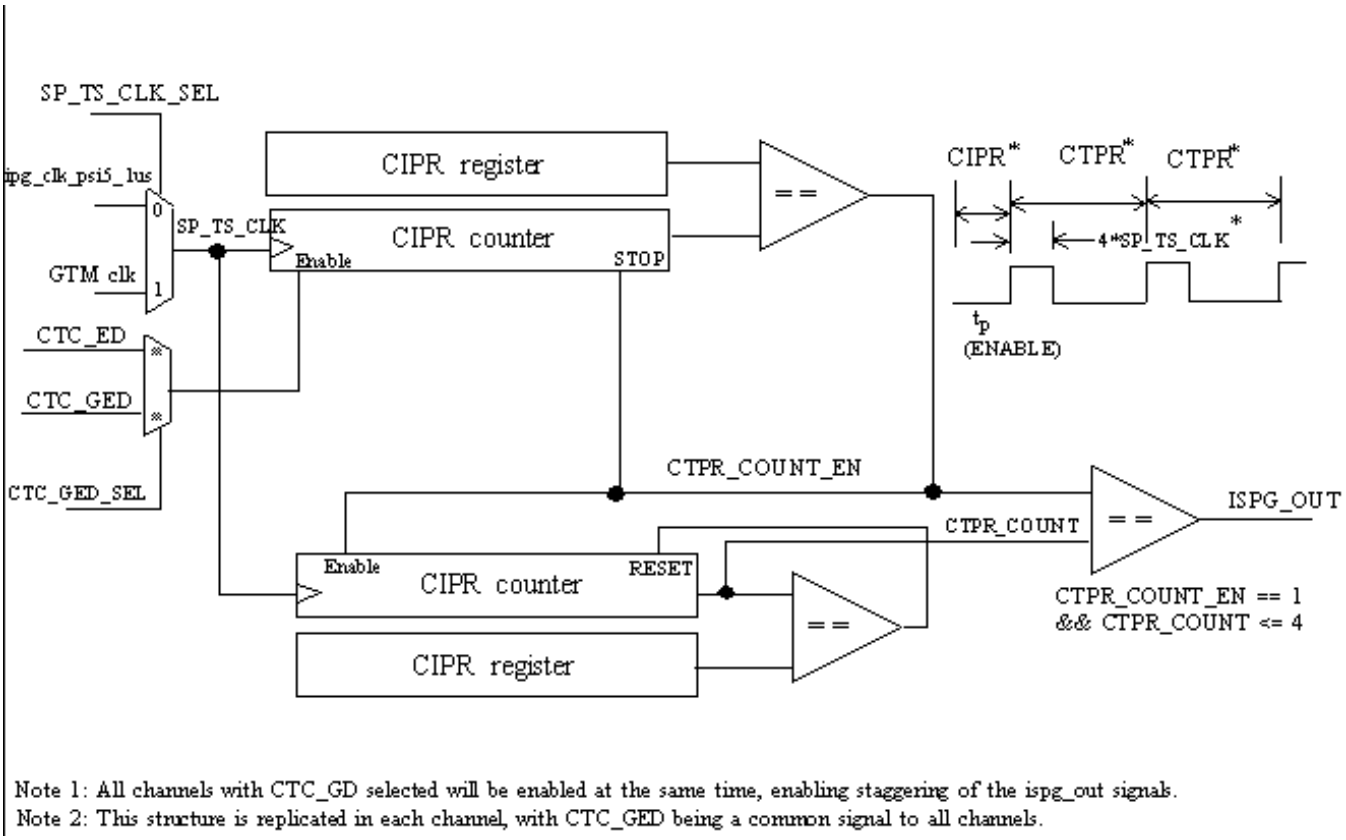


Figure 56-30. Sync Pulse Generation Parameters and Jitters

Figure 56-30 shows the various jitters.



Note 1: All channels with CTC_GD selected will be enabled at the same time, enabling staggering of the ispg_out signals.
Note 2: This structure is replicated in each channel, with CTC_GED being a common signal to all channels.

Figure 56-31. ISPG(Internal Sync Pulse Generator) Structure

56.7.2 Valid states for integrated sync pulse generator

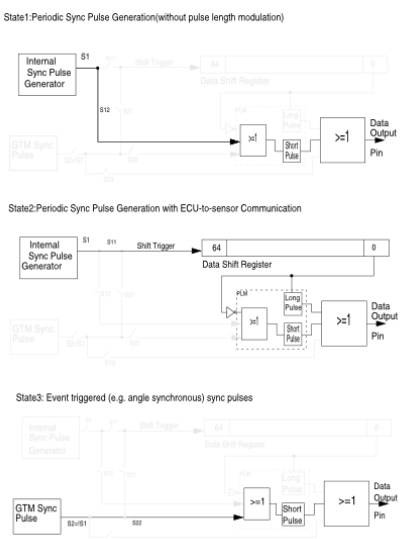


Figure 56-32. ECU to Sensor Communication States

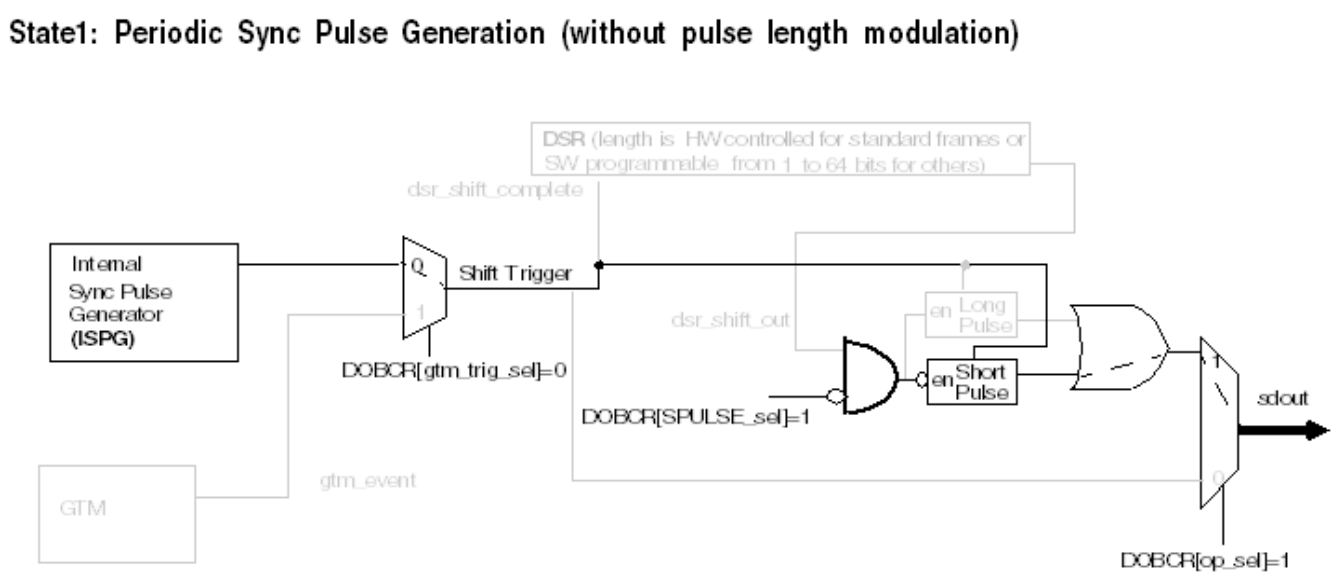


Figure 56-33. ECU to Sensor Communication States - State 1

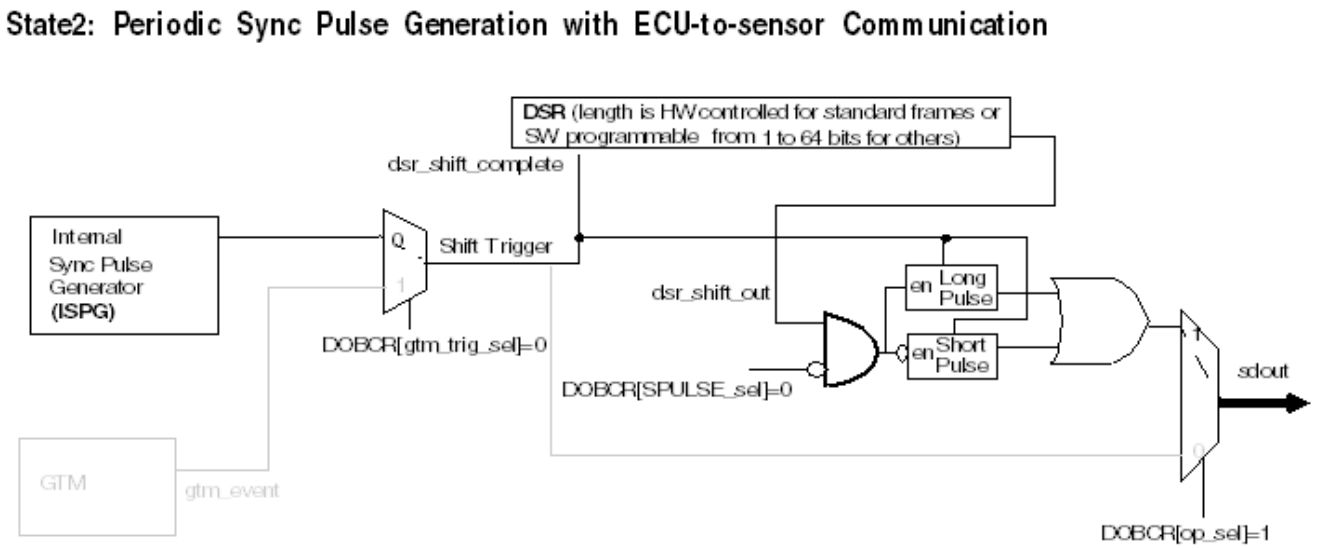


Figure 56-34. ECU to Sensor Communication States - State 2

State3: Event triggered (e.g. angle synchronous) sync pulses

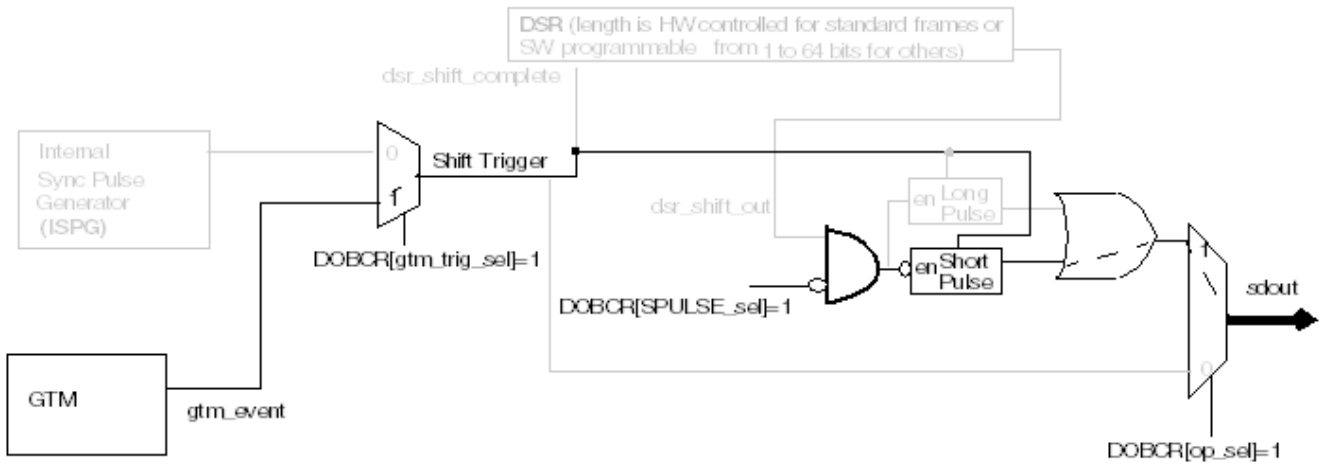


Figure 56-35. ECU to Sensor Communication States - State 3

State4: Event triggered sync pulse, including ECU-to-sensor communication

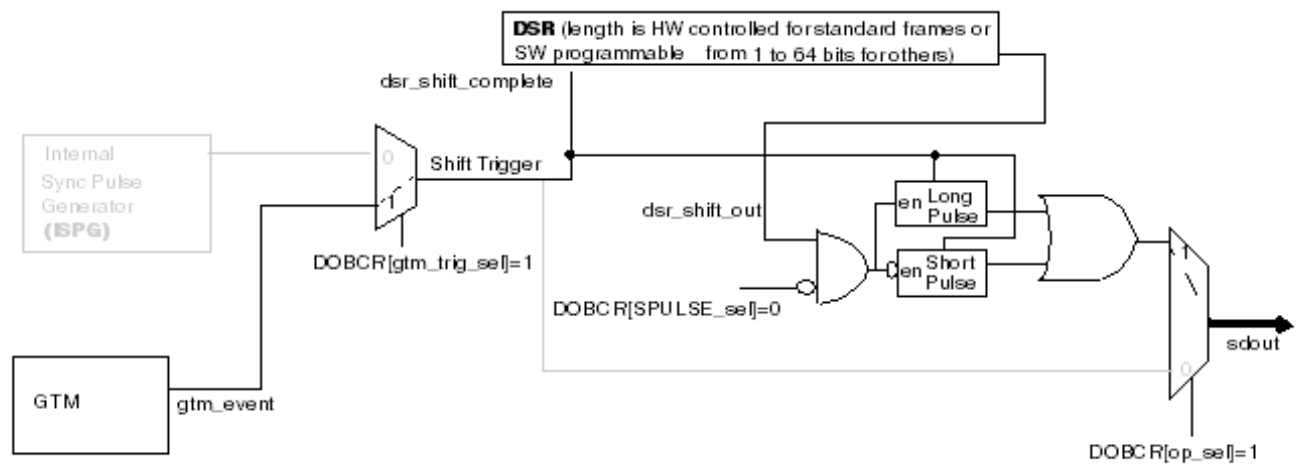


Figure 56-36. ECU to Sensor Communication States - State 4

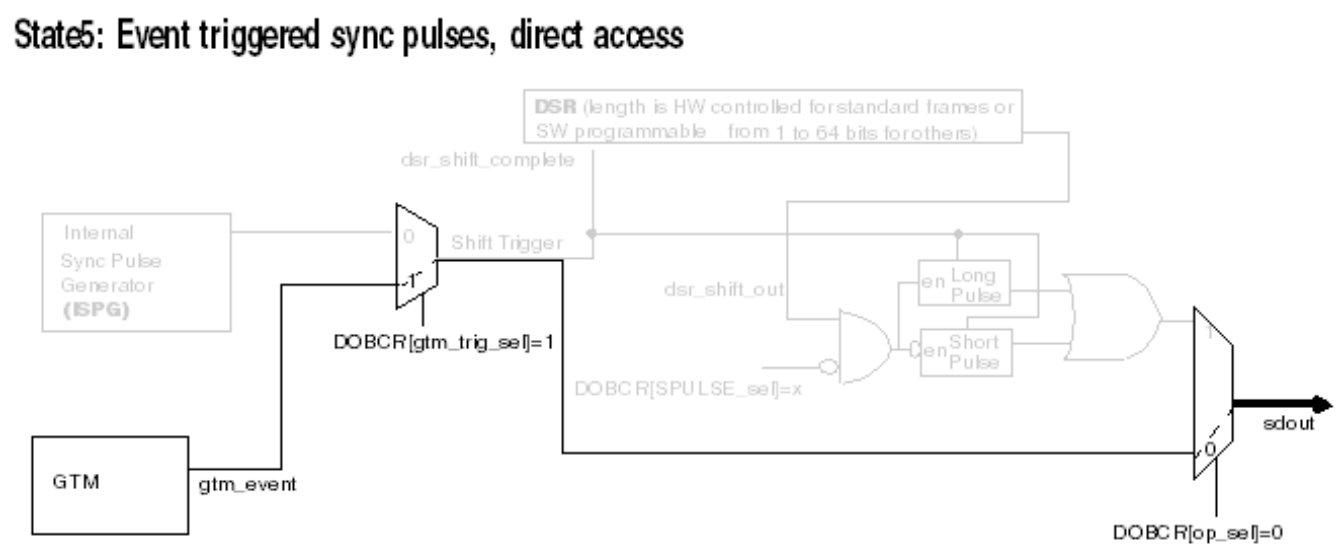


Figure 56-37. ECU to Sensor Communication States - State 5

56.7.3 Implementation of EI Diagnostic Register

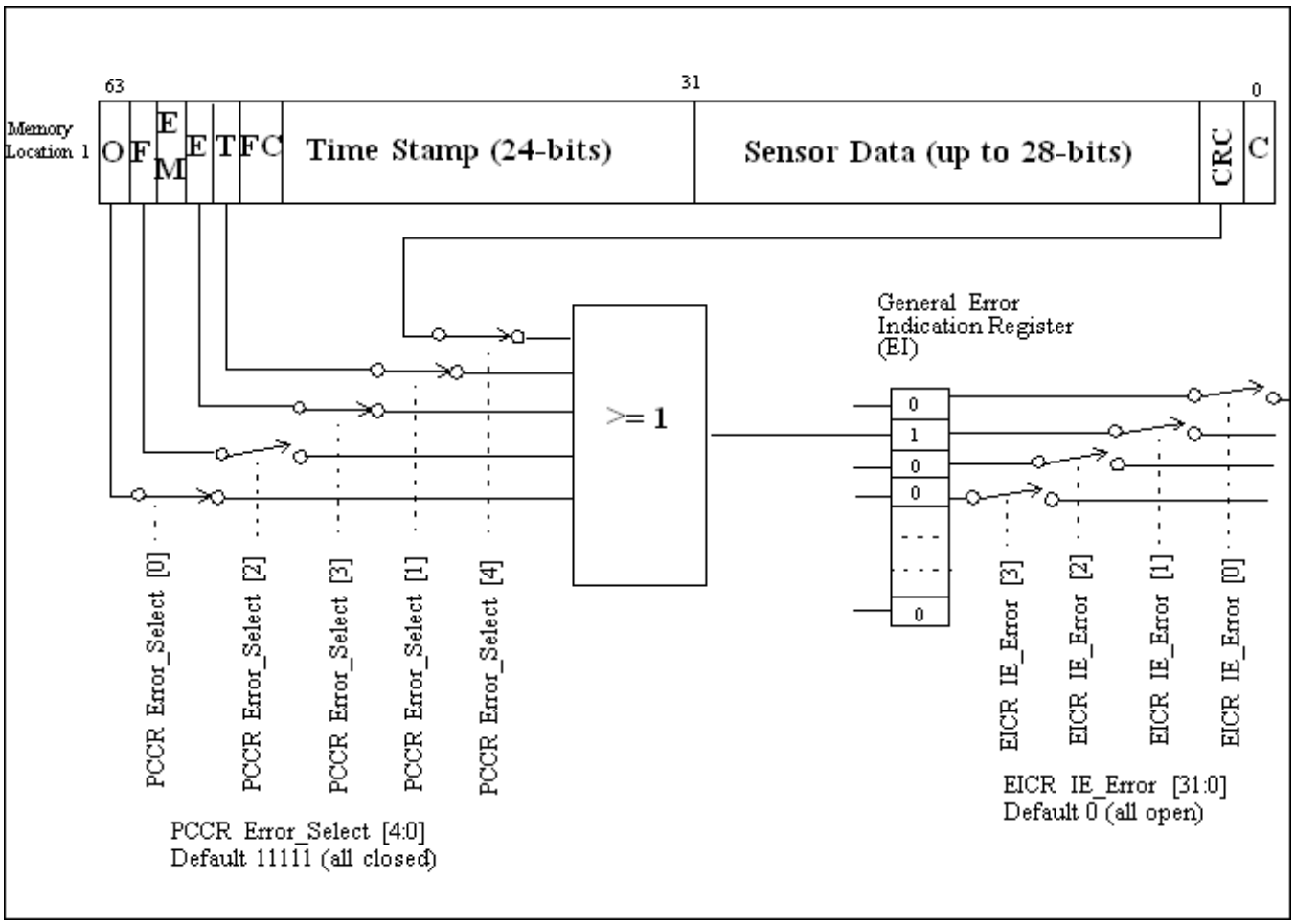


Figure 56-38. Implementation of EI Diagnostic Register

Chapter 57
Peripheral Sensor Interface-Support Module (PSI5-S)

57.1 Introduction

The PSI5 support IP (PSI5-S) module is used to interface directly to a UART compatible PSI5 transceiver over a high speed UART link. The transceiver is required to implement a compliant PSI5 module for all vehicle applications as is currently defined in the PSI5 protocol specification V2.0.

PSI5 messages from sensors received by the transceiver are decomposed into UART bytes and sent to the PSI5-S which in turn reconstructs the PSI5 messages from the UART packet, and makes them available for the application in system RAM.

ECU-to-sensor commands are sent from the PSI5-S module to the transceiver, which in turn generates the appropriate logical 1's and 0's in the SYNC pulses sent on the bus. The PSI5-S supports all command formats of the PSI5 V2.0 including the "tooth gap" method and "variable length" pulses.

At the system level, the PSI5-S supports one transceiver with up to seven PSI5 channels. Apart from these seven channels, there is an eighth channel available for special purposes, including direct communication with the transceiver and storage of corrupt messages.

57.1.1 Overview

PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make PSI5 also suitable for many other automotive sensor applications. The development goal of PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented license free.

The main features of PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications. The main features of the PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 kbit/s or optional 189 kbit/s
- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

The PSI5-S module can be configured to support the following PSI5 features:

- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

The PSI5-S module requires the physical and data link layer of the PSI5 to be implemented outside its interface. It also requires the PSI5 information to be embedded in the form of UART packet, from where it reconstructs the PSI5 message.

The module receives PSI5 message, from up to seven PSI5 channels each with upto six sequential frames. This is achieved via the external UART compliant transceiver. The PSI5 message in each of these frames are decomposed into three-to-six UART bytes (depending on PSI5 data payload) and streamed on a UART link at up to 6.25 Mbit/s to the PSI5-S module. In addition, via the PSI5-S interface, the application sends commands to the transceiver. The transceiver interprets them as either SYNC pulse commands (which will result in short, long or no SYNC pulses on the PSI5 bus), configuration commands (which will result in updates to the transceiver configuration) or diagnostic commands (which will result in the transceiver responding directly over the UART connection). In addition to the seven channels above there is an additional Channel, known as Channel0, which is used for special purposes like storing the direct responses from the transceiver or storing the unrecoverable messages.

The PSI5-S consists of three main functional blocks.

1. UART front end - Used as PSI5-S front end. When PSI5 functionality is not required then it is possible to use this as a standard UART (without LIN) also.
2. Message Recovery Unit - consisting of two state machines.
   - The first state machine interprets the input UART data stream and rebuilds the PSI5 message contained in the up to six UART bytes and also checks for the status of the correct reception of PSI5 message.
   - The second state machine sends SYNC pulse commands via the UART TxBuffer, to the external UART compliant transceiver, which in turn outputs SYNC pulses (short, long, or no SYNC pulse) on the PSI5 bus used to synchronize sensor responses in a bus configuration, and where required, also sends commands to sensors as defined in the PSI5 protocols V2.0.
3. PSI5 message interface - It is composed of two MRU buffers, MRU buffer1 and MRU buffer2. Both of them are collectively referred to as MRU output buffer. The MRU buffer1 is internal to the IP and not accessible from outside the IP. It is three word deep. MRU buffer2 is accessible to the application and is 4 word deep. MRU buffer2 can be read by the DMA interface, to transfer the stored PSI5 messages, to Channel specific mailboxes in system RAM directly.

Note

: General Description

- The location in system RAM where the messages are stored are referred to as the Mailbox. The mailbox is divided on the basis of channel and each channel is divided on the basis of slots on the PSI5 bus. Thus Mailbox1,location1 would mean Mailbox location for Channel1 and Slot No "1". This is also denoted by $MBOX_1$, LOC_1 or MBOX1, LOC1. This terminology would be followed in the subsequent sections of the doc.

  Each "location" of each mailbox channel is made up of 3x32 bits. The maximum size of each mailbox can be 6 "locations". There are 8 such mailboxes. For channels operating in asynchronous mode, the whole of the Mailbox would act like a ring buffer and all the 6 locations would be filled in that manner.

Mailbox0 only has two locations MBOX0,LOC0 and MBOX0,LOC1. MBOX0,LOC0 is reserved for messages intended for Channel0(used for transceiver specific messages) while MBOX0,LOC1 contain "unrecoverable" messages for any channel. Other 4 locations of the MBOX0 are reserved.

- Also a "UART byte" would mean the "byte" that is received by the UART receiver. "UART packet" would mean the group of UART bytes from which the PSI5 message would be extracted. "PSI5 message" means the message contained in the MRU buffers which contains the actual PSI5 message along with the timestamp and the error status bits.

Figure 57-1 shows the block diagram of the PSI5-S module.

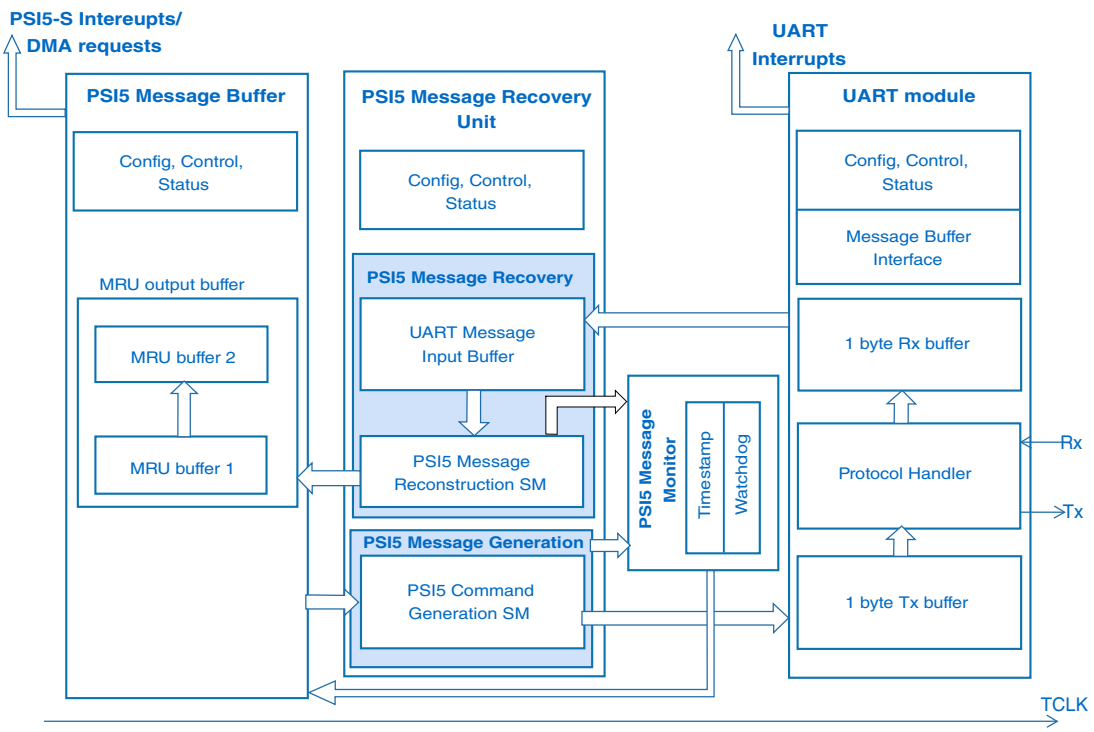


Figure 57-1. PSI5-S digital interface block diagram

57.1.2 Features

- IPS interface for register read/write (big endian).

- Support for up to 7 PSI5 channels(Channel1 to Channel7) with each channel supporting up to 6 different Frame Streams (Slots). Each of the Frame Streams can be individually configured for Parity/CRC, payload Length, Synchronous/ Asynchronous Operation, TimeStamp Capture event, Timestamp A or Timestamp B selection.
- Additional Channel0 for special functions.
- Supports bidirectional full duplex operations, via the UART Rx and Tx links.
- Provision of Extended CRC (XCRC) to check the coherence of the UART packet that comprise the PSI5 message.
- Message Reconstruction Unit (MRU) buffer2 (consisting of four 32-bit words). Three of these words contain the PSI5 messages (PSI5 message payload, Message Status flags, Time Stamp) and the fourth one contains the mailbox address to which the three PSI5 words need to be transferred.
- A 24-bit time stamp register, clocked with either the module clock or the GTM clock (gtm_trig). The time stamp appended to the PSI5 message in the MRU, can be configured to be, either the one that is captured when a new PSI5 message is detected, or the one that is captured when the Output ECU to Sensor command is transferred from the UART TX buffer to the UART Tx Shift register.
- IPS interface DMA support, for transferring the data in the MRU output buffer to system memory.
- Downstream Data Shift Register (DDSR) on per channel basis supporting the following:
    - Application can write the ECU to Sensor commands to the DDSR.
    - Supports automatic addition of the start bits/stuff bits /CRC calculation to the contents of DDSR.
    - Provision of resetting and rejecting the contents of DDSR at any stage of command transmission.
- Support for conversion of the logic levels contained in the DDSR, to UART commands to be sent to the transceiver.
- Provision of writing the Software calculated commands directly to the UART TX Buffer. This can be used for direct writing of the transceiver status and/or diagnostic commands.

- Configurable UART message idle time counter, that indicates to the PSI5 MRU when an idle time at least as long as the configured length (number of bit times), has occurred on the Rx line.
- Provision of configuring the PSI5-S front-end as a standalone UART module, when the PSI5-S module is disabled. In this mode the module, does not have any relation with the PSI5 protocol.
- Support of up to 6.25 Mbit/s UART Rx Baud Rate.
- An output clock for the PSI5 transceiver UART interface.

57.1.3 PSI5-S Global Modes of operation

The PSI5-S module can be configured in 4 different global modes of operation. These are UART_STDALONE, PS_DISABLE, PS_CONFIG and PS_NORMAL. These global modes can be entered by programming the PS_GLCR[GLOBAL_MODE] bits appropriately. Figure 57-2 shows the state transition diagram for these bits.

The PS_NORMAL,PS_CONFIG and PS_DISABLE are collectively referred to as the "PSI5-S" modes. Please see Figure 57-2. Once the PS_GLCR[GLOBAL_MODE] bits are appropriately configured for the "PSI5-S" modes, the information that the intended mode of operation is fully achieved is reflected by the setting of the PS_GLSR[GL_MODETR_DONE] status bit. An interrupt can also be generated for this event, in case the PS_GLCR[GL_MODETR_DONE_EN] bit is set to "1". Thus, only when PS_GLSR[GL_MODETR_DONE] goes to "1" it should be considered that the IP has entered the desired "PSI5-S" mode.

This bit is reset to "0" when the mode is changed from any of the PSI5-S modes to the UART_STDALONE mode.

Following are the descriptions of the different global modes.

57.1.3.1 Standalone UART mode (UART_STDALONE mode)

This is the default state after the module reset. In this mode, only the PS_GLCR[GLOBAL_MODE] register bits and the registers of the UART module are accessible for writing. This is the mode which is to be used when the PSI5-S front end has to be used as a standalone UART. This state indicates that the UART will now function as a standard UART and the PSI5-S core will be disabled. In this state, the PSI5-

S register space is not accessible (except PS_GLCR[GLOBAL_MODE] registers) and any access to the registers of PSI5-S module (except PS_GLCR[GLOBAL_MODE] registers and UART specific registers) will generate transfer error.

57.1.3.2 Disable mode (PS_DISABLE)

PS_Disable mode in the PSI5 reduces power consumption. In this mode the RX, TX and common time base counters are disabled for any operation. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Please see Table 57-1 for the details of these bits. Note that when entering PS_Disable mode from any other mode, the register contents are retained and are not changed. The time stamp counter is however reset when entering Disable mode.

In "PS_Disable" mode, only PS_GLCR[GLOBAL_MODE] register bits are accessible for read/write. Accessing any other register bits will generate transfer error.

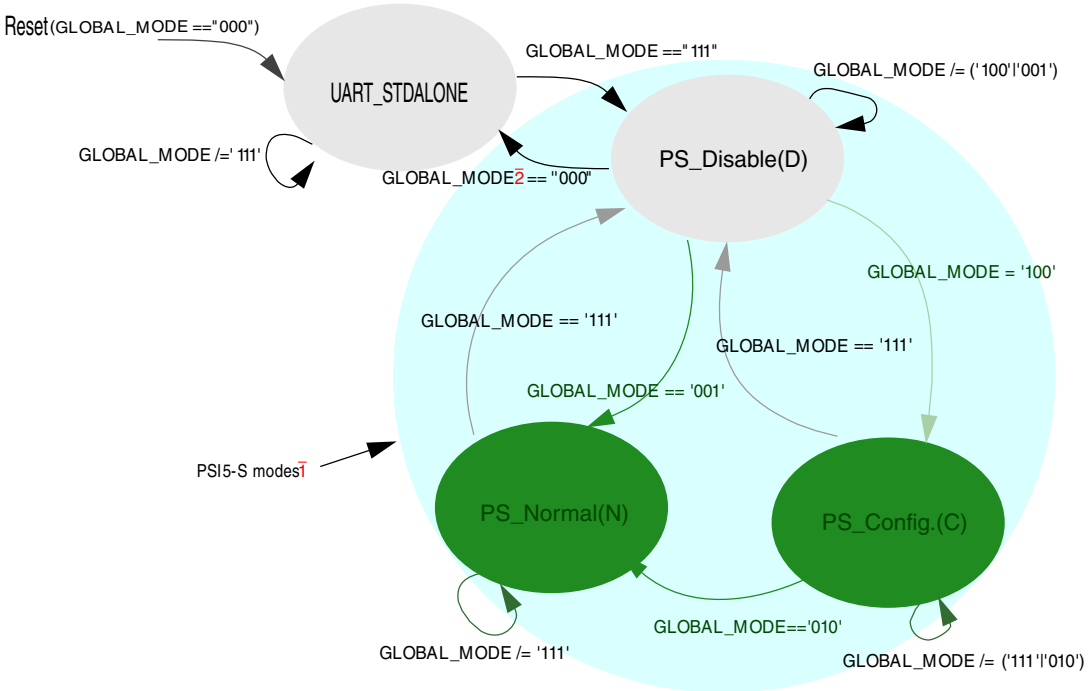
There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter "Configuration/Normal mode" based on the value of the PS_GLCR[GLOBAL_MODE] bits.

57.1.3.3 Configuration mode (PS_CONFIG mode)

In the configuration mode all PSI5 global, channel specific configuration/control registers and UART registers can be written. The receiver and transmitter sub-blocks remain disabled in this mode. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Please see Table 57-1 for the details of these bits.

57.1.3.4 Normal mode (PS_NORMAL mode)

In PS_NORMAL mode, the receiver and the transmitter blocks become active, error handling is enabled, UART starts its operation and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5 global and channel-specific configuration registers are locked for writing, although these registers can still be read. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Please see Table 57-1 for the details of these bits.Note that there are still some register bits that can be written in this mode also. Please refer to the detailed register description.



1 PSI5-S module can be configured in 4 different Global modes of operation . These are UART_STDALONE , PS_NORMAL,PS_CONFIG and PS_DISABLE . The PS_NORMAL,PS_CONFIG and PS_DISABLE are collectively referred to as “PSI5-S” modes.

2 PS_GLSR[GL_MODETR_DONE] status bit is reset to “0” when the transition occurs from PS_DISABLE to UART_STDALONE mode .

Figure 57-2. Operating modes

Table 57-1. State transition Table

Present State	PS_GLCR (State Variables)			Next State
	PS_GLCR[GLOBAL_MODE(2)]	PS_GLCR[GLOBAL_MODE(1)]	PS_GLCR[GLOBAL_MODE(0)]	
UART_STDALONE	1	1	1	PS_Disable
PS_Disable	0	0	0	UART_STDALONE
PS_Disable	0	0	1	PS_Normal
PS_Disable	1	0	0	PS_Config
PS_Config	1	1	1	PS_Disable
PS_Config	0	1	0	PS_Normal
PS_Normal	1	1	1	PS_Disable

57.1.4 PSI5-S System Modes of operation

Various modes of operation are discussed in the following sections.

57.1.4.1 Stop mode

For PSI5-S modes:

In this mode, upon the STOP request from the system level the IP generates a system level acknowledgement with the information that the clocks to the IP can be switched off. The process of the same is as described below.

At the PSI5-S Rx side, as soon as stop mode request is received by the IP then the current PSI5 message under construction (in the internal PSI5 message extraction logic) is immediately saved to the MRU output buffer1. Thus the assertion of the stop mode acts like a "forced" flushing for the current PSI5 message completion. Depending on the state of the PSI5 message under construction, the resultant PSI5 message might or might not have errors. Any byte that is being received by the UART is henceforth rejected. The internal PSI5 message extraction logic goes back to an "idle" state and remains so till the STOP mode request remains asserted.

At the PSI5-S Tx side, on the reception of the stop mode request, the internal sync pulse generation counters are stopped immediately (DDSR trigger counters) and no further command bits are shifted onto the UART Tx side. The UART Tx completes the current byte that is being transmitted and enters the idle state. No further transmissions take place.On deassertion of the stop mode, once the clocks restarts, the transmission of the commands resumes from where they had halted.

Once both the above processes are complete the IP generates acknowledgement for switching of the clocks to the PSI5-S IP.

For UART STDALONE mode:

For this please refer to the separate documentation describing the standalone LINFLEX module.

57.1.4.2 Debug mode

The IP enters this mode on system request.Any register can be read or written to by the debugger. The IP can enter the debug mode in addition to it being in any of the other modes (Uart Standalone/Disable/Normal/Config). In this mode the hardware clearing of the register PS_WDGTSSR remains disabled and any read of this register has no effect on the contents of this register.Apart from this there is no other difference as to whether the IP is in debug mode or not in the debug mode of operation.

57.2 External signal description

S.No	Name	Function	I/O	Pull-up
	Signals to/from pads			
1.	ipp_ind_linrx	LINUART receive pin. Used for Receiving UART bytes from transceiver which contain information to enable the rebuilding of the PSI5 message within the packet of between three and six UART bytes	I	Active
2.	ipp_do_lintx	LINUART transmit pin. Used for transmitting command bytes from TxFIFO to transceiver for generation of SYNC pulses (long, short or no) on the PSI5 bus or direct diagnostic/ status commands for the transceiver itself	O	Passive
3.	ipp_port_en_tx	LINUART transmission port enable:Connected to the enable pin of the pad which is to be used for "LINTX" functionlaity. When "1" it indicates that the IP is doing transmit of data.	O	Passive
4.	ipp_psi5_uart_tclk	Transceiver clock output, which drives the UART interface on the transceiver. The clock output frequency is configurable between 4 and 8 x UART baud rate on the interface between PSI5-S and transceiver.Frequency can be controlled at the system level through the Clock Generator Module (MC_CGM). The duty cycle of the transceiver clock is between 40% and 60%	O	Passive
IPS Interface				
Clocks and Reset				
49.	ipg_clk_ps_en	This is the clock enable signal that is used to gate the IP input, "ipg_clk_ps", at the system level. When "1", it indicates to the system that the "ipg_clk_ps" should be available. In stop mode this signal retains its previous state.	O	

57.3 Transfer Error Generation

The IP generates transfer error cases in a number of scenarios:

1. Trying to write to any register in the PS_DISABLE mode (apart from PS_GLCR and PS_GLSR registers) will result in the generation of the transfer error.
2. Trying to write to any "Read Only" registers (those which have "R, R, R" in all the D, C, N columns in the memory map table in Memory map and register description) will generate a transfer error in any of the PS_DISABLE, PS_NORMAL and PS_CONFIG modes. Apart from the above “Read Only” registers , trying to write to

any register bits outside there intended mode of operation(Normal/Config) would be ignored without the generation of any transfer error. For bit level operation and mode access of each register, please see UART register configuration and PSI5-S Register description in Memory map and register description section.

3. Trying to access (R/W) any register space, which are declared as reserved in the table in Memory map and register description section or which is unimplemented, will generate a transfer error.
4. Reading of any register in any mode and if that register is NOT in unimplemented/ reserved space, then a transfer error will not be generated.
5. Trying to write to any PSI5-S register (starting from locations 0x00B4, but apart from PS_GLCR and PS_GLSR registers) in the UART_STANDALONE mode, will generate a transfer error.
6. In UART Standalone mode, the UART behaves as a standalone IP and for detailed transfer error cases in this mode, the UART standalone documentation should be referred to.

57.4 Memory map and register description

UART register configuration

This section contains information of the UART registers (registers with address offset FBF7_4000 to FBF7_405C) that are specifically required for correct functioning of the IP in the “PSI5-S” modes.

In order to operate the PSI5-S module, the UART configuration registers need to be programmed with appropriate values. The values that are required for different register fields are defined in the following register descriptions. Some bits have to be programmed with definite values of “1” or “0” only. This configuration has to be strictly followed for a predictable operation ofthe PSI5-S core. All these bits should be programmed in the PS_CONFIG mode before enabling the UART RX or the TX paths.

In the following UART register description “PS_NOUSE_BIT” against a bit name and bit description, means that the specific register bit is present in the UART module but it is NOT REQUIRED in the PSI5-S modes of operation. For correct operation in the PSI5-S modes , the corresponding register bit might be required to be programmed as “1” or a “0” ; the same would be mentioned in the bit description.

Note that a bit with “PS_NOUSE_BIT” against its description would still be required in the UART STANDALONE mode. For detailed information of the UART register bit functioning in the UART STANDLONE mode, please refer to the separate document describing the standalone LINUART module.There are some other register bits which

dont have a “PS_NOUSE_BIT” against them. These can be programmed as required. A bit mentioned as “Reserved” would mean that neither it is available in the UART STANDLONE mode nor in the PSI5-S modes.

PSI5-S Register description

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode.

PS_GLCR[GLOBAL_MODE] are the only register bits that can be written in the PS_DISABLE mode.

All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to write to “read only” bits of implemented register space, will be ignored without the generation of any error.

The registers with address offset FBF7_40B4 onwards are under this category of registers.

NOTE

In the "PSI5S Memory Map" table below, R/W = Read/Write, R = Read-only, W = Write-only, W1C = write-1-to-clear and W1S = “write-to-1-shot wherein writing a 1 causes an effect on the register bit while writing a 0 has no effect ; a read always returning a 0”. The access shown in this table are for Disable mode. For knowing access for all the registers in Config mode and Normal mode, please refer to the Register Access in Different Modes.

NOTE

All 32-bit registers are word/half word/byte addressable for write. Read is always word addressable.

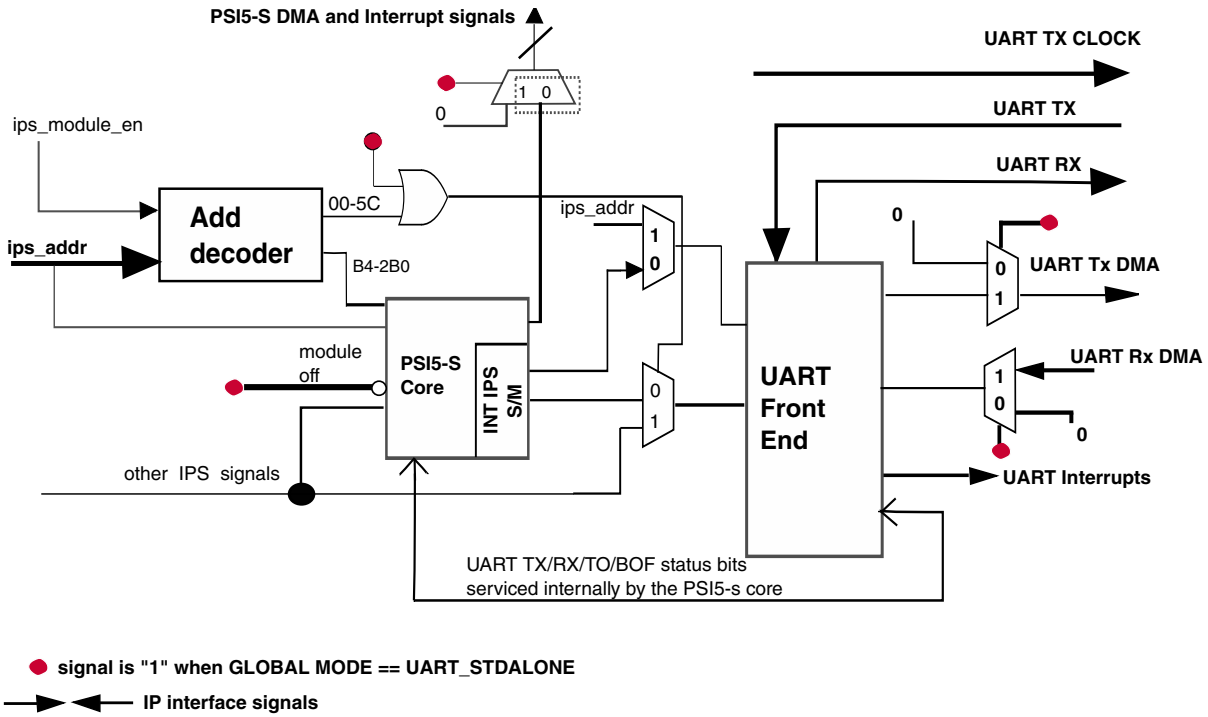


Figure 57-3. LINUART interaction with the PSI5-S core

PSI5S memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PSI5-S LIN Control Register 1 (PSI5S_LINCR1)	32	R/W	0000_0092h	57.4.1/2979
4	PSI5-S LIN Interrupt enable register (PSI5S_LINIER)	32	R/W	0000_0000h	57.4.2/2980
8	PSI5-S LIN Status Register (PSI5S_LINSR)	32	R	0000_0040h	57.4.3/2983
10	PSI5-S UART Mode Control Register (PSI5S_UARTCR)	32	R/W	0000_0000h	57.4.4/2984
14	PSI5-S UART Mode Status Register (PSI5S_UARTSR)	32	w1c	0000_0000h	57.4.5/2987
24	PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR)	32	R/W	0000_0000h	57.4.6/2989
28	PSI5-S LIN Integer Baud Rate Register (PSI5S_LINIBRR)	32	R/W	0000_0000h	57.4.7/2990
30	PSI5-S LIN Control Register 2 (PSI5S_LINCR2)	32	R	0000_4000h	57.4.8/2992
38	PSI5-S Buffer Data Register Least Significant (PSI5S_BDRL)	32	R/W	0000_0000h	57.4.9/2993
3C	PSI5-S Buffer Data Register Most Significant (PSI5S_BDRM)	32	R/W	0000_0000h	57.4.10/ 2993
4C	PSI5-S Global Control register (PSI5S_GCR)	32	R	0000_0000h	57.4.11/ 2994
50	PSI5-S UART Preset Timeout Register (PSI5S_UARTPTO)	32	R/W	0000_0FFFh	57.4.12/ 2995
54	UPSI5-S ART Current Timeout register (PSI5S_UARTCTO)	32	R	0000_0000h	57.4.13/ 2996
58	DMA Tx Enable Register (PSI5S_DMATXE)	32	R/W	0000_0000h	57.4.14/ 2997

Table continues on the next page...

PSI5S memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5C	DMA Rx Enable Register (PSI5S_DMARXE)	32	R/W	0000_0000h	57.4.15/ 2998
60	PSI5-S UART Tx Idle Delay Time Register (PSI5S_PTD)	32	R/W	0000_0000h	57.4.16/ 2999
B4	PSI5-S Global Control Register (PSI5S_GLCR)	32	R/W	0000_0000h	57.4.17/ 3000
B8	PSI5-S Global Status Register (PSI5S_GLSR)	32	R	0010_0000h	57.4.18/ 3004
BC	PSI5-S CHANNEL_BASE_ADDRESS (PSI5S_CH_BASE_ADDR)	32	R	0000_0000h	57.4.19/ 3005
C0	PSI5-S MRU OUTPUT BUFFER2 REGISTER0 (PSI5S_MRU_BUF2_REG0)	32	R	0000_0000h	57.4.20/ 3006
C4	PSI5-S MRU OUTPUT BUFFER2 REGISTER1 (PSI5S_MRU_BUF2_REG1)	32	R	0000_0000h	57.4.21/ 3007
C8	PSI5-S MRU OUTPUT BUFFER2 REGISTER2 (PSI5S_MRU_BUF2_REG2)	32	R	0000_0000h	57.4.22/ 3009
CC	PSI5-S MRU OUTPUT BUFFER2 REGISTER3 (PSI5S_MRU_BUF2_REG3)	32	R	0000_0000h	57.4.23/ 3010
E0	PSI5-S Mbox Status Irq (PSI5S_MBOX_SR_IRQ)	32	R	0000_0000h	57.4.24/ 3011
E4	PSI5-S Error Status Irq (PSI5S_ERR_SR_IRQ)	32	w1c	0000_0000h	57.4.25/ 3013
E8	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ0)	32	R/W	0000_0000h	57.4.26/ 3017
EC	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ0)	32	R/W	0000_0000h	57.4.27/ 3018
F0	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ1)	32	R/W	0000_0000h	57.4.26/ 3017
F4	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ1)	32	R/W	0000_0000h	57.4.27/ 3018
F8	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ2)	32	R/W	0000_0000h	57.4.26/ 3017
FC	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ2)	32	R/W	0000_0000h	57.4.27/ 3018
100	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ3)	32	R/W	0000_0000h	57.4.26/ 3017
104	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ3)	32	R/W	0000_0000h	57.4.27/ 3018
108	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ4)	32	R/W	0000_0000h	57.4.26/ 3017
10C	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ4)	32	R/W	0000_0000h	57.4.27/ 3018
110	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ5)	32	R/W	0000_0000h	57.4.26/ 3017

Table continues on the next page...

PSI5S memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
114	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ5)	32	R/W	0000_0000h	57.4.27/ 3018
118	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ6)	32	R/W	0000_0000h	57.4.26/ 3017
11C	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ6)	32	R/W	0000_0000h	57.4.27/ 3018
120	PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ7)	32	R/W	0000_0000h	57.4.26/ 3017
124	PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ7)	32	R/W	0000_0000h	57.4.27/ 3018
128	PSI5-S Watchdog Error Status and Watchdog Timestamp status register (PSI5S_WDGTSSR)	32	R	0000_0000h	57.4.28/ 3020
12C	PSI5-S ECU to Sensor Direct Command Write register (PSI5S_DIRCMD)	32	R/W	0000_0000h	57.4.29/ 3021
16C	PSI5-S channel 0 message configuration register A (PSI5S_MSGA_CH0)	32	R/W	0000_0051h	57.4.30/ 3022
170	PSI5-S channel 0 message configuration register B (PSI5S_MSGB_CH0)	32	R/W	0000_0008h	57.4.31/ 3023
178	PSI5-S Mailbox status register channel0 (PSI5S_MBOX_SR_CH0)	32	w1c	0000_0000h	57.4.32/ 3024
190	PSI5-S channel message configuration register A (PSI5S_MSGA_CH1)	32	R/W	0000_0000h	57.4.33/ 3026
194	PSI5-S channel message configuration register B (PSI5S_MSGB_CH1)	32	R/W	1084_2108h	57.4.34/ 3030
19C	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH1)	32	w1c	0000_0000h	57.4.35/ 3032
1A0	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH1)	32	R/W	0000_0000h	57.4.36/ 3036
1A4	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH1)	32	R/W	0000_0000h	57.4.37/ 3037
1A8	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH1)	32	R/W	0000_0000h	57.4.38/ 3038
1AC	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH1)	32	R/W	0000_0800h	57.4.39/ 3038
1B0	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH1)	32	w1c	0000_0020h	57.4.40/ 3042
1B4	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH1)	32	R	0000_0000h	57.4.41/ 3043
1B8	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH1)	32	R/W	0000_0000h	57.4.42/ 3044
1CC	PSI5-S channel message configuration register A (PSI5S_MSGA_CH2)	32	R/W	0000_0000h	57.4.33/ 3026
1D0	PSI5-S channel message configuration register B (PSI5S_MSGB_CH2)	32	R/W	1084_2108h	57.4.34/ 3030

Table continues on the next page...

PSI5S memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1D8	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH2)	32	w1c	0000_0000h	57.4.35/ 3032
1DC	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH2)	32	R/W	0000_0000h	57.4.36/ 3036
1E0	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH2)	32	R/W	0000_0000h	57.4.37/ 3037
1E4	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH2)	32	R/W	0000_0000h	57.4.38/ 3038
1E8	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH2)	32	R/W	0000_0800h	57.4.39/ 3038
1EC	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH2)	32	w1c	0000_0020h	57.4.40/ 3042
1F0	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH2)	32	R	0000_0000h	57.4.41/ 3043
1F4	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH2)	32	R/W	0000_0000h	57.4.42/ 3044
208	PSI5-S channel message configuration register A (PSI5S_MSGA_CH3)	32	R/W	0000_0000h	57.4.33/ 3026
20C	PSI5-S channel message configuration register B (PSI5S_MSGB_CH3)	32	R/W	1084_2108h	57.4.34/ 3030
214	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH3)	32	w1c	0000_0000h	57.4.35/ 3032
218	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH3)	32	R/W	0000_0000h	57.4.36/ 3036
21C	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH3)	32	R/W	0000_0000h	57.4.37/ 3037
220	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH3)	32	R/W	0000_0000h	57.4.38/ 3038
224	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH3)	32	R/W	0000_0800h	57.4.39/ 3038
228	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH3)	32	w1c	0000_0020h	57.4.40/ 3042
22C	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH3)	32	R	0000_0000h	57.4.41/ 3043
230	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH3)	32	R/W	0000_0000h	57.4.42/ 3044
244	PSI5-S channel message configuration register A (PSI5S_MSGA_CH4)	32	R/W	0000_0000h	57.4.33/ 3026
248	PSI5-S channel message configuration register B (PSI5S_MSGB_CH4)	32	R/W	1084_2108h	57.4.34/ 3030
250	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH4)	32	w1c	0000_0000h	57.4.35/ 3032
254	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH4)	32	R/W	0000_0000h	57.4.36/ 3036

Table continues on the next page...

PSI5S memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
258	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH4)	32	R/W	0000_0000h	57.4.37/ 3037
25C	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH4)	32	R/W	0000_0000h	57.4.38/ 3038
260	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH4)	32	R/W	0000_0800h	57.4.39/ 3038
264	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH4)	32	w1c	0000_0020h	57.4.40/ 3042
268	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH4)	32	R	0000_0000h	57.4.41/ 3043
26C	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH4)	32	R/W	0000_0000h	57.4.42/ 3044
280	PSI5-S channel message configuration register A (PSI5S_MSGA_CH5)	32	R/W	0000_0000h	57.4.33/ 3026
284	PSI5-S channel message configuration register B (PSI5S_MSGB_CH5)	32	R/W	1084_2108h	57.4.34/ 3030
28C	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH5)	32	w1c	0000_0000h	57.4.35/ 3032
290	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH5)	32	R/W	0000_0000h	57.4.36/ 3036
294	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH5)	32	R/W	0000_0000h	57.4.37/ 3037
298	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH5)	32	R/W	0000_0000h	57.4.38/ 3038
29C	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH5)	32	R/W	0000_0800h	57.4.39/ 3038
2A0	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH5)	32	w1c	0000_0020h	57.4.40/ 3042
2A4	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH5)	32	R	0000_0000h	57.4.41/ 3043
2A8	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH5)	32	R/W	0000_0000h	57.4.42/ 3044
2BC	PSI5-S channel message configuration register A (PSI5S_MSGA_CH6)	32	R/W	0000_0000h	57.4.33/ 3026
2C0	PSI5-S channel message configuration register B (PSI5S_MSGB_CH6)	32	R/W	1084_2108h	57.4.34/ 3030
2C8	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH6)	32	w1c	0000_0000h	57.4.35/ 3032
2CC	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH6)	32	R/W	0000_0000h	57.4.36/ 3036
2D0	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH6)	32	R/W	0000_0000h	57.4.37/ 3037
2D4	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH6)	32	R/W	0000_0000h	57.4.38/ 3038

Table continues on the next page...

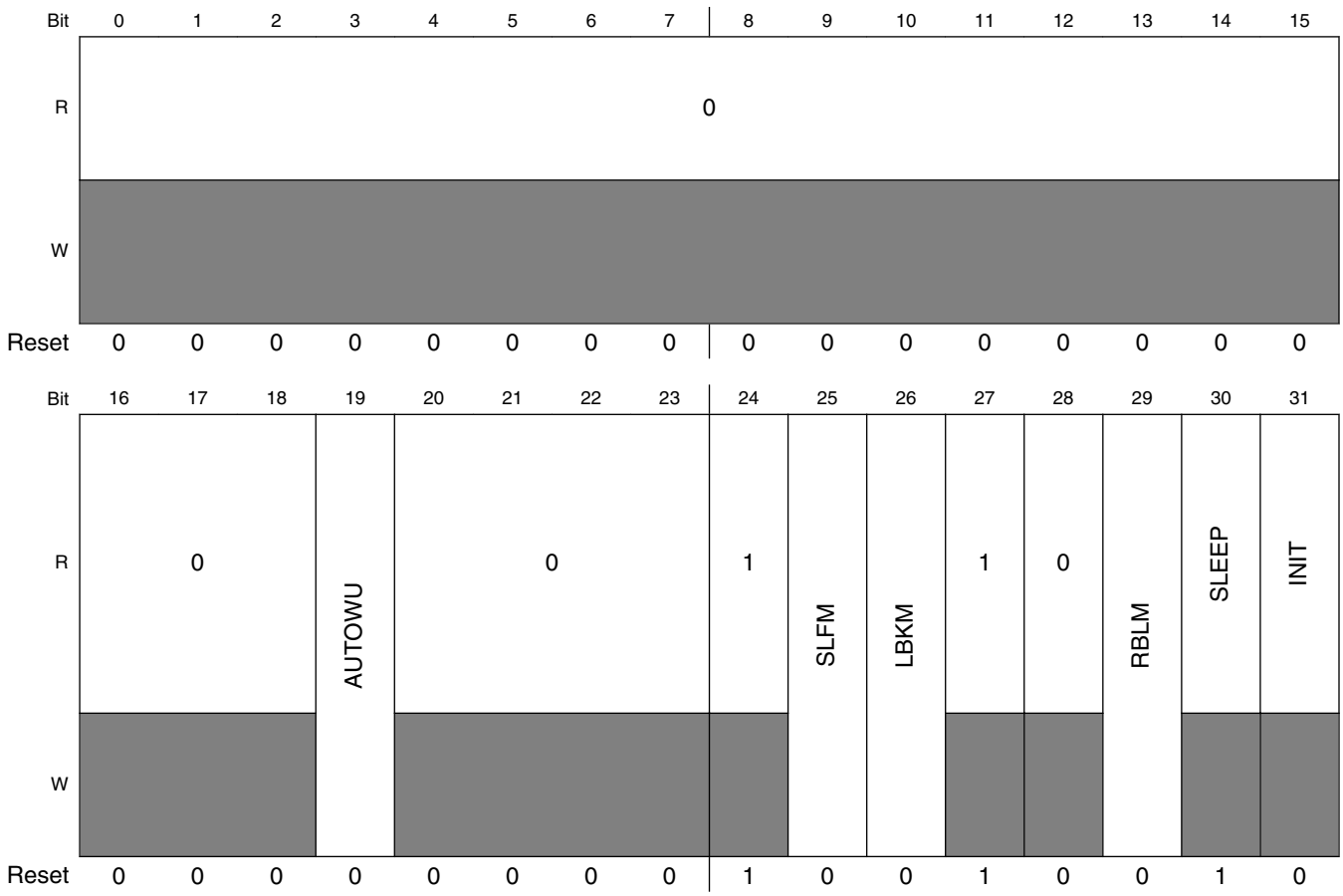
PSI5S memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2D8	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH6)	32	R/W	0000_0800h	57.4.39/ 3038
2DC	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH6)	32	w1c	0000_0020h	57.4.40/ 3042
2E0	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH6)	32	R	0000_0000h	57.4.41/ 3043
2E4	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH6)	32	R/W	0000_0000h	57.4.42/ 3044
2F8	PSI5-S channel message configuration register A (PSI5S_MSGA_CH7)	32	R/W	0000_0000h	57.4.33/ 3026
2FC	PSI5-S channel message configuration register B (PSI5S_MSGB_CH7)	32	R/W	1084_2108h	57.4.34/ 3030
304	PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH7)	32	w1c	0000_0000h	57.4.35/ 3032
308	PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH7)	32	R/W	0000_0000h	57.4.36/ 3036
30C	PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH7)	32	R/W	0000_0000h	57.4.37/ 3037
310	PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH7)	32	R/W	0000_0000h	57.4.38/ 3038
314	PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH7)	32	R/W	0000_0800h	57.4.39/ 3038
318	PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH7)	32	w1c	0000_0020h	57.4.40/ 3042
31C	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH7)	32	R	0000_0000h	57.4.41/ 3043
320	PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH7)	32	R/W	0000_0000h	57.4.42/ 3044

57.4.1 PSI5-S LIN Control Register 1 (PSI5S_LINCR1)

The PS_LINCR1 register contains control bits used to configure features of the PSI5-S UART Front end.

Address: FBF7_4000h base + 0h offset = FBF7_4000h



PSI5S_LINCR1 field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 AUTOWU	PS_NOUSE_BIT Always program as "0" in PSI5-S modes.
20–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

Table continues on the next page...

PSI5S_LINCR1 field descriptions (continued)

Field	Description
25 SLFM	Self test mode To be used along with the LBKM bit. When this bit is set then in addition to the Loop Back mode ; the "ipp_do_lintx" is driven to "1" . 0 Self test mode disabled 1 Self test mode enabled
26 LBKM	Loop Back mode Programmed in PS_CONFIG mode. When set then then the "ipp_do_lintx" output is internally looped back and connected to "ipp_ind_linrx" pin . 0 Loop Back Mode disabled 1 Loop Back mode enabled
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 RBLM	PS_NOUSE_BIT Always program as 0 in PSI5-S modes.
30 SLEEP	PS_NOUSE_BIT **NOTE:** This bit is Read Only Bit in PSI5-S modes and R/W in UART Standlone mode. **NOTE:** This bit is read as “1” in PS_DISABLE mode but is read as “0” in PS_NORMAL and PS_CONFIG mode.
31 INIT	PS_NOUSE_BIT **NOTE:** This bit is Read Only Bit in PSI5-S modes and R/W in UART Standlone mode. **NOTE:** This bit is read as “0” in PS_DISABLE and PS_NORMAL mode , and “1” in PS_CONFIG mode.

57.4.2 PSI5-S LIN Interrupt enable register (PSI5S_LINIER)

Address: FBF7_4000h base + 4h offset = FBF7_4004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	0					FEIE	BOIE	0	WUIE	0	TOIE	DRIE	DTIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_LINIER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZIE	Stuck at Zero Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core **NOTE:** This bit is also writeable in PS_NORMAL mode. 0 No interrupt 1 Interrupt generation enabled
17 OCIE	PS_NOUSE_BIT Always program as 0 in PSI5-S modes. **NOTE:** This bit is also writeable in PS_NORMAL mode.
18–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEIE	Frame Error Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (FEF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core **NOTE:** This bit is also writeable in PS_NORMAL mode. 0 No interrupt 1 Interrupt generation enabled
24 BOIE	Buffer Overrun Error Interrupt Enable An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core **NOTE:** This bit is also writeable in PS_NORMAL mode. 0 No interrupt 1 Interrupt generation enabled
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 WUIE	PS_NOUSE_BIT Always program as 0 in PSI5-S modes. **NOTE:** This bit is also writeable in PS_NORMAL mode.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TOIE	Timeout Interrupt Enable An interrupt is generated if this bit is set and the Timeout Flag (TO) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core **NOTE:** This bit is also writeable in PS_NORMAL mode.

Table continues on the next page...

PSI5S_LINIER field descriptions (continued)

Field	Description
	0 No interrupt 1 Interrupt generation enabled.
29 DRIE	Data Reception complete Interrupt enable An interrupt is generated if this bit is set and the Data Recieved Flag (DRF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core **NOTE:** This bit is also writeable in PS_NORMAL mode. 0 No interrupt 1 Interrupt generation enabled
30 DTIE	Data Transmitted Interrupt enable An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in PS_UARTSR is set. This interrupt can be used by the Application for debugging when it wants to read the byte that is under transmission. This byte can be read from the PS_BDRL[DATA_TX[0]] location. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **NOTE:** This bit is also writeable in PS_NORMAL mode. 0 No interrupt 1 Interrupt generation enabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.3 PSI5-S LIN Status Register (PSI5S_LINSR)

This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes. For PSI5-S modes, always program with 0x0000_00000 in Config Mode. For Normal and Disable Mode, please refer to "Registers Access in Different Modes" table.

Address: FBF7_4000h base + 8h offset = FBF7_4008h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS3	LINS2	LINS1	LINS0	0					1	0					
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

PSI5S_LINSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LINS3	PS_NOUSE_BITS. Read Value is "dont care" in PSI5-S modes
17 LINS2	PS_NOUSE_BITS. Read Value is "dont care" in PSI5-S modes
18 LINS1	PS_NOUSE_BITS. Read Value is "dont care" in PSI5-S modes

Table continues on the next page...

PSI5S_LINSR field descriptions (continued)

Field	Description
19 LINS0	PS_NOUSE_BITS. Read Value is "dont care" in PSI5-S modes
20–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.4 PSI5-S UART Mode Control Register (PSI5S_UARTCR)

Table 57-2. OSR and CSP values

OSR	CSP
4	2, 3
5	2, 3, 4
6	3, 4, 5
8	NA

Address: FBF7_4000h base + 10h offset = FBF7_4010h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS	CSP			OSR				ROSE	NEF			PCE_TXDTU	SBUR		WLS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFLTFC			RDFLRFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE_Rx	WL0	UART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_UARTCR field descriptions

Field	Description
0 MIS	PS_NOUSE_BIT Always program as "1" in PSI5-S modes.

Table continues on the next page...

PSI5S_UARTCR field descriptions (continued)

Field	Description
1–3 CSP	Configurable Sample Point These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the range of values shown in Table 57-2 for a certain over sampling rate. For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.
4–7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper funtionality, OSR can only take values 4, 5, 6 and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.
8 ROSE	Reduced Over Sampling Enable The default value of this bit is 0. 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate.
9–11 NEF	PS_NOUSE_BIT Always program as "000" in PSI5-S modes.
12 PCE_TXDTU	In PSI5-S mode Parity Control Enable for Tx path(In UART STANDLONE mode used to disable UART timeout-Refer to LINUART stanalone documentation for more details) Can be programmed in PS_CONFIG mode only and when PS_UARTCR[0] == 1. When programmed as "1" then a 1 bit additional parity bit is added to the transmitted 8 bits of UART data, so a total of 9 bits including partity are present between a start and a stop bit , in the transmitted data When programmed as "0" then total of 8bits data is present between a start and a stop bit in the transmitted data. 0 Parity generation for Tx path disabled. 1 Parity generation for Tx path enabled.
13–14 SBUR	PS_NOUSE_BIT Always program as "00" in PSI5-S modes.
15 WLS	PS_NOUSE_BIT Always program as "0" in PSI5-S modes.
16–18 TDFLTFC	PS_NOUSE_BIT Always program as "000" in PSI5-S modes.
19–21 RDFLRFC	PS_NOUSE_BIT Always program as "000" in PSI5-S modes.
22 RFBM	PS_NOUSE_BIT Always program as "0" in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
23 TFBM	PS_NOUSE_BIT Always program as "0" in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
24 WL1	PS_NOUSE_BIT

Table continues on the next page...

PSI5S_UARTCR field descriptions (continued)

Field	Description
	Always program as "0" in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
25 PC1	Parity Control {PC1,PC0} Can be programmed only in the PS_CONFIG mode only and when PS_UARTCR[UART] == 1 . 00 Parity is Even for both the Rx/Tx paths 01 Parity is Odd for both the Rx/Tx paths. 10 A logical 0 is taken as parity bit for both the Rx and the Tx paths 11 A logical 1 is taken as parity bit for both the Rx and the Tx paths.
26 RxEn	Receiver Enable This bit can be programmed only when PS_UARTCR[0] == 1. 0 Receiver disabled 1 Receiver enabled
27 TxEn	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when when PS_UARTCR[UART] == 1
28 PC0	Parity Control {PC1,PC0} Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1 . 00 Parity is Even for both the Rx/Tx paths 01 Parity is Odd for both the Rx/Tx paths. 10 A logical 0 is taken as parity bit for both the Rx and the Tx paths 11 A logical 1 is taken as parity bit for both the Rx and the Tx paths.
29 PCE_Rx	Recieve Parity check enable. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1. When programmed as "1" then a 1 bit additional parity bit is expected along with the recieved 8 bits of UART data, so a total of 9 bits including partity are expected between a start and a stop bit , in the recieved data. When programmed as "0" then total of 8bits data is expected between a start and a stop bit in the receieved data. 0 Disable the parity checking on the Rx path 1 Enable the parity checking on the Rx path
30 WL0	PS_NOUSE_BIT Always program as "1" in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
31 UART	PS_NOUSE_BIT Always program as "1" in PSI5-S modes.

Table continues on the next page...

PSI5S_UARTCR field descriptions (continued)

Field	Description
	Can be programmed in PS_CONFIG mode only

57.4.5 PSI5-S UART Mode Status Register (PSI5S_UARTSR)

Address: FBF7_4000h base + 14h offset = FBF7_4014h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF	DTF	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_UARTSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated on the UART interrupt lines if the SZIE bit in PS_LINIER is set.
17 OCF	PS_NOUSE_BIT Read Only Bit.
18 PE3	PS_NOUSE_BIT Read Only Bit.
19 PE2	PS_NOUSE_BIT Read Only Bit.
20 PE1	PS_NOUSE_BIT Read Only Bit.
21 PE0	Parity Error flag for Data0 byte These bits indicate if there is a Parity Error in the recieved byte. No interrupt is generated if this error occurs. This flag is automatically cleared by hardware. The software can poll this flag for debugging purposes. 0 No parity error 1 Parity error in the corresponding received byte
22 RMB	PS_NOUSE_BIT Read Only Bit.

Table continues on the next page...

PSI5S_UARTSR field descriptions (continued)

Field	Description
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). FEF is automatically cleared by hardware. An interrupt is generated if this bit is set and the FEIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and the previous byte has not yet been read from the UART Rx buffer due to some fault in PSI5-S core reading logic. It is automatically cleared by hardware. An interrupt is generated if this bit is set and the BOIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	PS_NOUSE_BIT Read Only Bit.
27 RFNE	PS_NOUSE_BIT Read Only Bit.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs - in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO is automatically cleared by hardware. An interrupt is generated if this bit is set and the TOIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
29 DRF	Data Reception Completed Flag DRF is set by hardware in UART buffer mode and indicates that one byte has been recieved and is ready for being read. DRF is automatically cleared by hardware. An interrupt is generated if this bit is set and the DRIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
30 DTF	Data Transmission Completed Flag DTF is set by hardware in UART buffer mode and indicates that data transmission is completed. DTF is automatically cleared by hardware. When the SW desires to write the commands directly to the UART Tx buffer for transmission then it can monitor this bit for an indication of when the UART Tx buffer is ready to accept the next 8 bits of the data. An interrupt will be generated if the DTIE bit in PS_LINIER is set.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

57.4.6 PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

The UART baud rate is programmed in two registers: the LIN Integer Baud Rate register(PS_LINIBRR) and the LIN Fraction Baud Rate register(PS_LINFBRR). The baud rate registers can be programmed only during the PS_CONFIG mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

TxBR = RxBR = ipg_baud_clk/ (16 x LDIV) (when UARTCR[ROSE] == 0)

TxBR = RxBR = ipg_baud_clk/UARTCR[OSR] x LINIBRR (when UARTCR[ROSE] == 1)

LDIV is an unsigned fixed point number which comprises the mantissa and the fraction. The mantissa is coded into 20 bits of LINIBRR , and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled(UARTCR[ROSE] ==1) then PS_LINFBRR is not used. Hence, LDIV contains only the integer part programmed in LINIBRR. Following are 2 examples for two different baud rate calculations.

Example 1 :

With UARTCR[ROSE] = 0.

LDIV = 1041.67, ipg_baud_clk = 80 MHz

LINIBRR = 1041d = 411 h

LINFBRR = 11d = B h

TxBR = RxBR = 80 MHz/(16 x 1041.67) = 4.8 Kbit/s

Example 2 :

When UARTCR[ROSE] = 1 :

LDIV = 10 d, LIN_CLK = 80 MHz

LINIBRR = 10 d, UARCR[OSR] = 4

TxBR = RxBR = ipg_baud_clk/ (UART[OSR] x LDIV) = 80 MHz / (4 x 10) = 2 Mbit/s

Table 57-3. LINFBRR and fraction (LDIV) values

LINFBRR[4:0]	Fraction (LDIV)
0h	0
1h	1/16
...	...
Eh	14/16
Fh	15/16

Address: FBF7_4000h base + 24h offset = FBF7_4024h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																												FBR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_LINFBRR field descriptions

Field	Description
0–27 Reserved	This field is reserved. Reserved Read returns 0.
28–31 FBR	Fractional Baud rates. See Table 57-3.

57.4.7 PSI5-S LIN Integer Baud Rate Register (PSI5S_LINIBRR)

This register consists of control bits that decide the baud rate along with the PS_LINFBRR. It can be programmed only in Initialization mode.

The UART baud rate is programmed in two registers: the LIN Integer Baud Rate register(PS_LINIBRR) and the LIN Fraction Baud Rate register(PS_LINFBRR). The baud rate registers can be programmed only during the PS_CONFIG mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

TxBR = RxBR = ipg_baud_clk/ (16 x LDIV) (when UARTCR[ROSE] == 0)

TxBR = RxBR = ipg_baud_clk/UARTCR[OSR] x LINIBRR (when UARTCR[ROSE] == 1)

LDIV is an unsigned fixed point number which comprises of the mantissa and the fraction. The mantissa is coded into 20 bits of LINIBRR , and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled(UARTCR[ROSE] ==1) then PS_LINFBRR is not used. Hence LDIV contains only the integer part programmed in LINIBRR. Following are 2 examples for two different baud rate calculations.

Example 1 :

With UARTCR[ROSE] = 0.

LDIV = 1041.67, ipg_baud_clk = 80 MHz

LINIBRR = 1041d = 411 h

LINFBRR = 11d = B h

TxBR = RxBR = 80 Mhz/(16 x 1041,67) = 4.8 Kbit/s

Example 2 :

When UARTCR[ROSE] = 1 :

LDIV = 10 d, LIN_CLK = 80 MHz

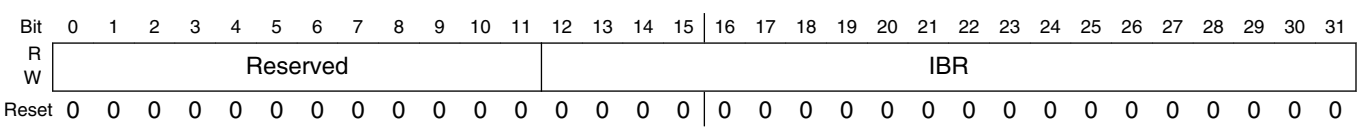
LINIBRR = 10 d, UARCR[OSR] = 4

TxBR = RxBR = ipg_baud_clk/ (UART[OSR] x LDIV) = 80 MHz / (4 x 10) = 2 Mbit/s

Table 57-4. LINIBRR and mantissa (LDIV) values

LINIBRR[19:0]	Mantissa (LDIV)
00000h	LIN clock disabled
00001h	1
...	...
FFFFEh	1048574
FFFFFh	1048575

Address: FBF7_4000h base + 28h offset = FBF7_4028h



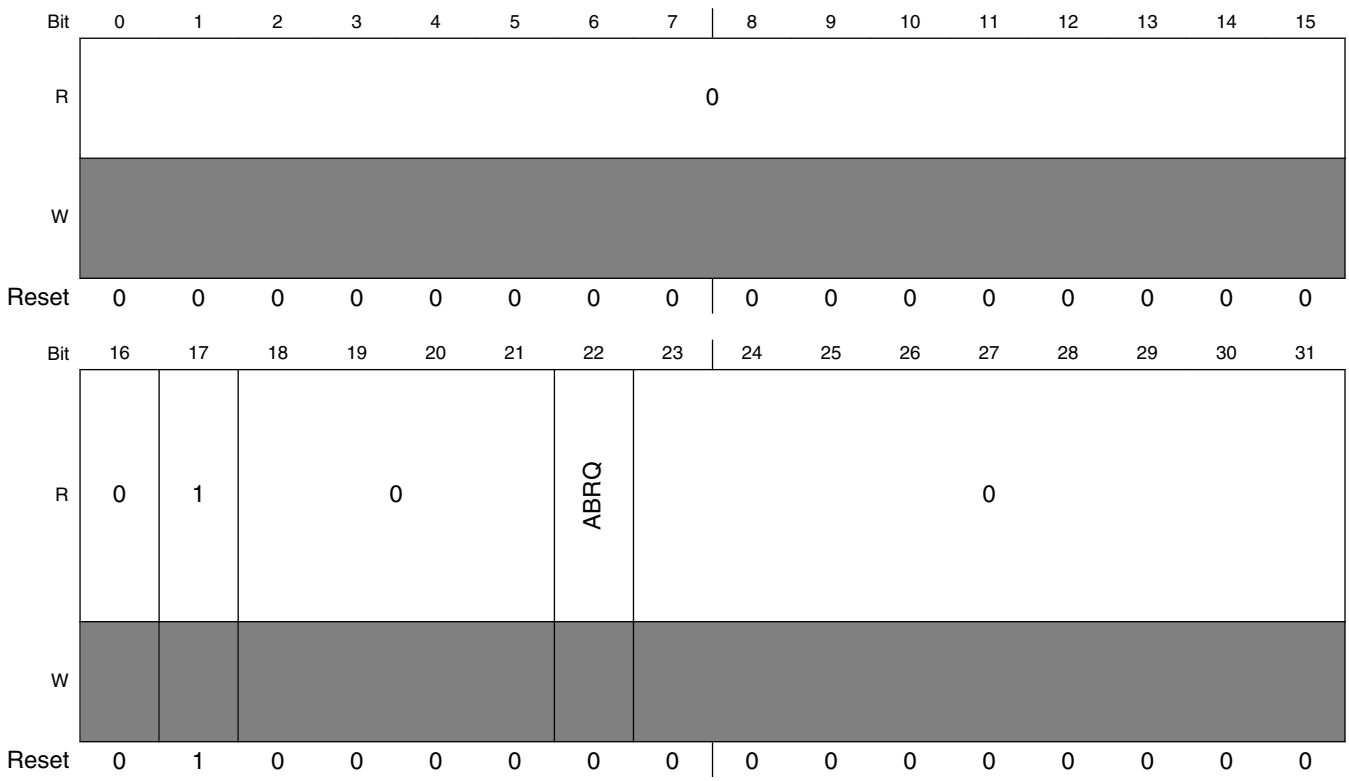
PSI5S_LINIBRR field descriptions

Field	Description
0–11 Reserved	This field is reserved.
12–31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. See Table 57-4. They can be read in any mode, and written only in PS_CONFIG mode.

57.4.8 PSI5-S LIN Control Register 2 (PSI5S_LINCR2)

This register includes control status bits related to buffer operations.

Address: FBF7_4000h base + 30h offset = FBF7_4030h



PSI5S_LINCR2 field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
18–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 ABRQ	Abort Request **NOTE:** Register bit can be read/set by software. Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. Transmission is aborted at the end of the current bit. **NOTE:** This bit is also writeable in PS_NORMAL mode.

Table continues on the next page...

PSI5S_LINCR2 field descriptions (continued)

Field	Description
23–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.9 PSI5-S Buffer Data Register Least Significant (PSI5S_BDRL)

Address: FBF7_4000h base + 38h offset = FBF7_4038h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_TX3								DATA_TX2								DATA_TX1								DATA_TX0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_BDRL field descriptions

Field	Description
0–7 DATA_TX3	PS_NOUSE_BIT Writing has no effect in fully configured PSI5-S mode. (All UART control registers are programmed as mentioned in this document).
8–15 DATA_TX2	PS_NOUSE_BIT Writing has no effect in fully configured PSI5-S mode. (All UART control registers are programmed as mentioned in this document).
16–23 DATA_TX1	PS_NOUSE_BIT Writing has no effect in fully configured PSI5-S mode. (All UART control registers are programmed as mentioned in this document).
24–31 DATA_TX0	Data Byte for Tx. In fully configured PSI5-S modes, the application has to take care to NEVER write to this register else the transmission of the UART can become unpredictable. This register can, however, be read by the application for debugging purpose since it gives the byte that has been written by the PSI5-S core to the UART, for transmission.

57.4.10 PSI5-S Buffer Data Register Most Significant (PSI5S_BDRM)

Address: FBF7_4000h base + 3Ch offset = FBF7_403Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_RX3								DATA_RX2								DATA_RX1								DATA_RX0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_BDRM field descriptions

Field	Description
0–7 DATA_RX3	PS_NOUSE_BIT

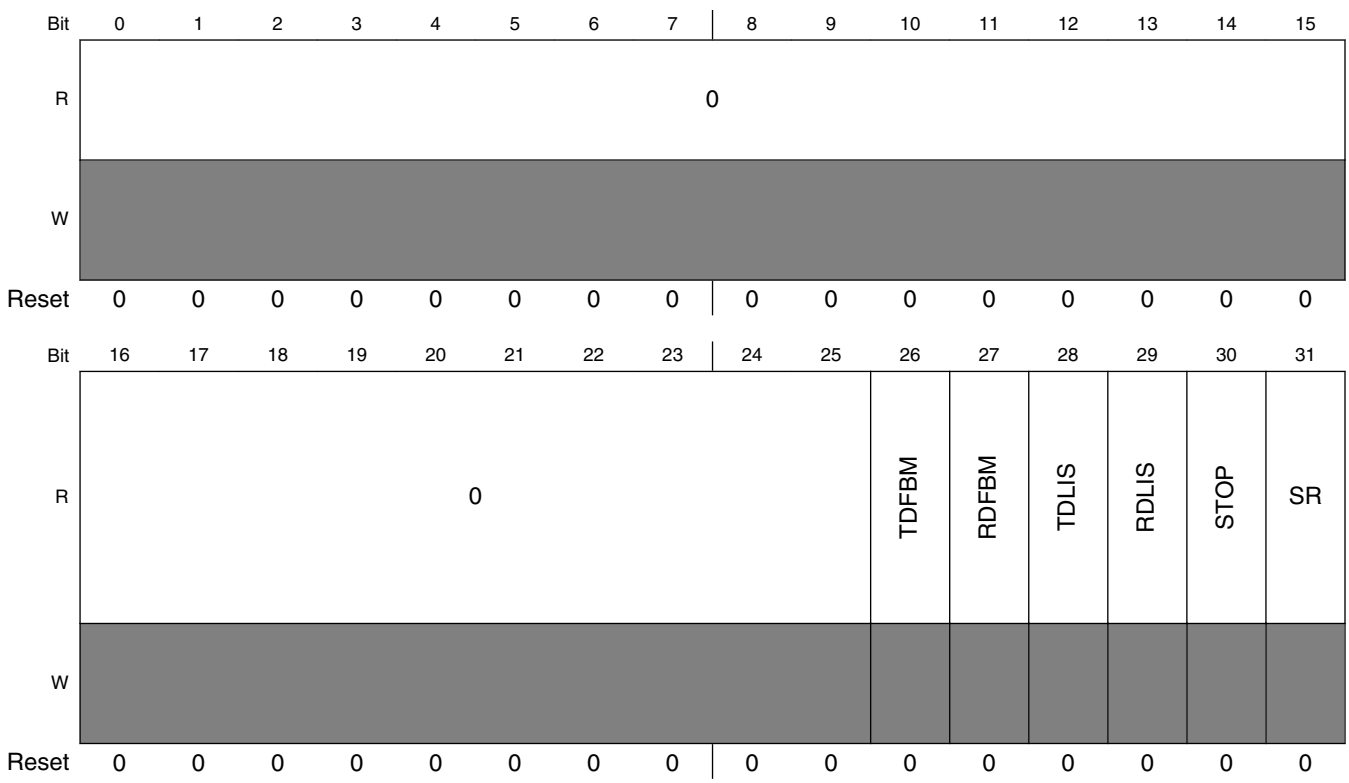
Table continues on the next page...

PSI5S_BDRM field descriptions (continued)

Field	Description
	Always read as "0" in fully configured PSI5-S mode. (All UART control registers are programmed as mentioned in this document.)
8–15 DATA_RX2	PS_NOUSE_BIT Always read as "0" in fully configured PSI5-S mode. (All UART control registers are programmed as mentioned in this document).
16–23 DATA_RX1	PS_NOUSE_BIT Always read as "0" in fully configured PSI5-S mode. (All UART control registers are programmed as mentioned in this document).
24–31 DATA_RX0	Recieve data byte for Rx Recieve data byte of the data field. This register can be read by the application for debugging purpose since it gives the byte that has been recieved by the UART core and will be send to the PSI5-S core.

57.4.11 PSI5-S Global Control register (PSI5S_GCR)

Address: FBF7_4000h base + 4Ch offset = FBF7_404Ch



PSI5S_GCR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

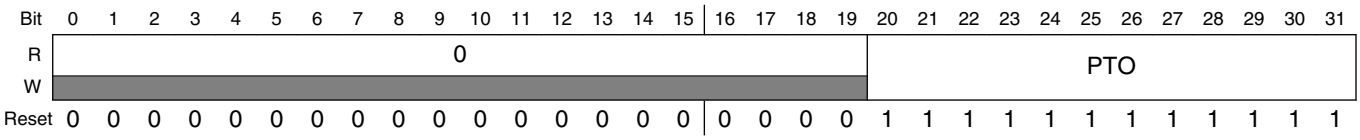
PSI5S_GCR field descriptions (continued)

Field	Description
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB : Register bit can be read in any mode, written only in PS_CONFIG mode. 0 The first bit of transmitted data is LSB - in other words, the first bit received is mapped on LSB bit (of DATA_TX in PS_BDRL register) 1 The first bit of received data is MSB - in other words, first bit received is mapped on MSB bit (of DATA_TX in PS_BDRL register)
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB : Register bit can be read in any mode, written only in PS_CONFIG mode. 0 The first bit of transmitted data is LSB - in other words, the first bit received is mapped on LSB bit (of DATA_RX in PS_BDRM register) 1 The first bit of received data is MSB - in other words, first bit received is mapped on MSB bit (of DATA_RX in PS_BDRM register)
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only). Register bit can be read in any mode, written only in PS_CONFIG mode. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) . Register bit can be read in any mode, written only in PS_CONFIG mode. 0 Received data is not inverted 1 Received data is inverted
30 STOP	PS_NOUSE_BIT Always program as "0" in PSI5-S modes.
31 SR	Soft reset SR executes a soft reset of the UART controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0. Register bit can be written only by software in PS_CONFIG mode. Bit is always read 0 by software.

57.4.12 PSI5-S UART Preset Timeout Register (PSI5S_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed to monitor the idle state of the reception line. This register can be written by software any time.

Address: FBF7_4000h base + 50h offset = FBF7_4050h



PSI5S_UARTPTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. The register must be programmed to a value of "Idle Timeout required in terms of number of bits recieved " at that particular baud rate. Refers also to register UARTCTO.

57.4.13 UPSI5-S ART Current Timeout register (PSI5S_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see Section 1.3.3.12, PSI5-S UART Preset Timeout Register (PS_UARTPTO) to monitor the idle state of the reception line.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with baud clock frequency of TxBR/RxBR. Also refer NOTE which follows Section 1.3.3.7, LIN Integer Baud Rate Register (PS_LINIBRR).
- Is automatically enabled when UARTCR[RXEN] = 1

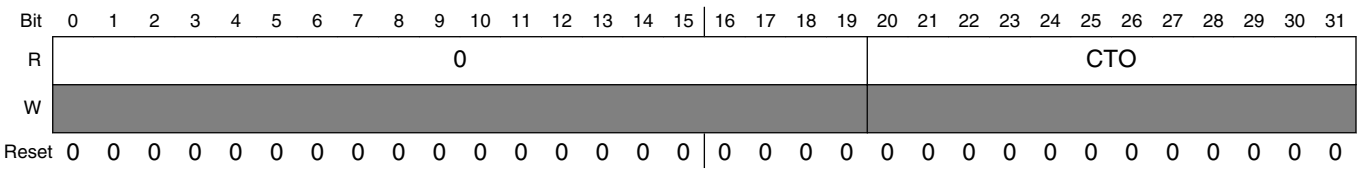
NOTE

Due to internal synchronization there is a delay between 4 to 6 clock cycles of “ipg_clk_ps”, for the internal counter’s value (which is clocked with baud frequency of TxBR/RxBR) to reflect on to UARTCTO

UARTCTO is reset when:

- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written.
- When the PSI5-S goes in PS_DISABLE mode.

Address: FBF7_4000h base + 54h offset = FBF7_4054h



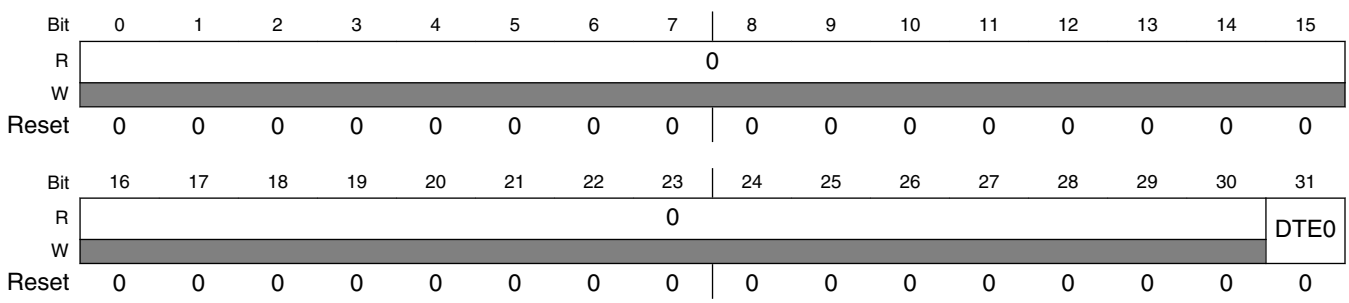
PSI5S_UARTCTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 CTO	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

57.4.14 DMA Tx Enable Register (PSI5S_DMATXE)

This register enables the DMA TX interface. This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes. For PSI5-S modes, always program with 0x0000_00000 in Config Mode. For Normal and Disable Mode, please refer to "Registers Access in Different Modes" table.

Address: FBF7_4000h base + 58h offset = FBF7_4058h



PSI5S_DMATXE field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 DTE0	PS_NOUSE_BIT Always program with "0" in PSI5-S modes

57.4.15 DMA Rx Enable Register (PSI5S_DMARXE)

This register enables the DMA RX interface. This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes. For PSI5-S modes, always program with 0x0000_00000 in Config Mode. For Normal and Disable Mode, please refer to "Registers Access in Different Modes" table.

Address: FBF7_4000h base + 5Ch offset = FBF7_405Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															DRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_DMARXE field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 DRE0	PS_NOUSE_BIT Always program with "0" in PSI5-S modes

57.4.16 PSI5-S UART Tx Idle Delay Time Register (PSI5S_PTD)

This register is used to add a programmable idle time between the data that is transmitted on the UART Tx line. The duration of this delay is programmable between 1 to 16 bit times.The programmable delay is such that when the normal ECU to Sensor communication(no direct command) is happening , then it comes into play between each byte of transmitted data , When the ECU to Sensor direct command is used then this delay comes into play between each direct command. The direct commands can be programmed as 1,2 and 4 bytes depending on the PS_GLCR[DIRCMD_LEN] register. For more information please see the SectionPSI5-S ECU to Sensor Direct Command Write register (PSI5S_DIRCMD) and SectionPSI5-S Global Control Register (PSI5S_GLCR) details.

Address: FBF7_4000h base + 60h offset = FBF7_4060h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											IFD				EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

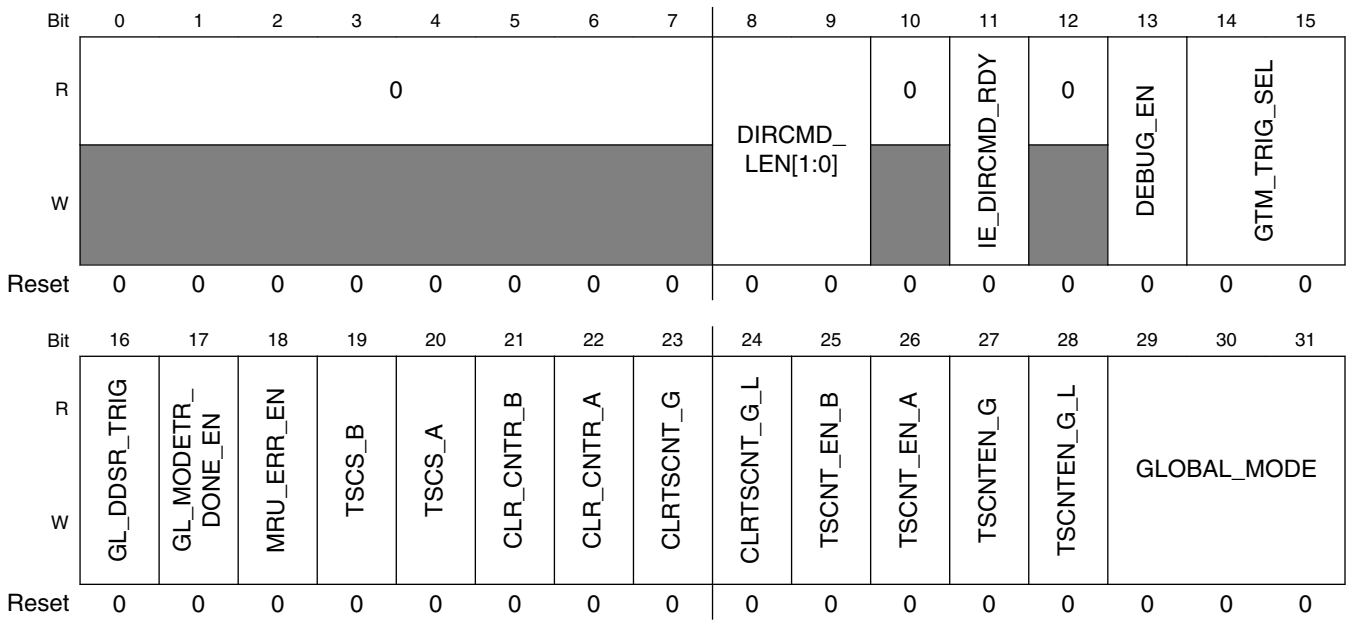
PSI5S_PTD field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 IFD	Interframe Delay. The idle time added between the UART data on the TX line is IFD+1 bit times. The UART data can be sized to be of either one byte or 1,2 and 4 bytes.
31 EN	IFD Enable 0 The IFD is active in the UART Tx path. 1 The IFD is inactive.

57.4.17 PSI5-S Global Control Register (PSI5S_GLCR)

The Global Control Register (PS_GLCR) provides control functions for all the PSI5-S channels simultaneously.

Address: FBF7_4000h base + B4h offset = FBF7_40B4h



PSI5S_GLCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–9 DIRCMD_ LEN[1:0]	Direct Command Length Register. This defines the length of the Direct Command that is transferred to the UART Tx line. The Direct command is written to the PS_DIRCMD register in the format {DIRCMD_BYTE3, DIRCMD_BYTE2, DIRCMD_BYTE1, DIRCMD_BYTE0}. Following is the bit description : **NOTE:** The idle time between the commands is still governed by the PS_PTD register in the LINUART module. **NOTE:** This register is also writeable in the NORMAL mode. 00 1 byte command : 1 byte {DIRCMD_BYTE0} of the PS_DIRCMD is transferred to the UART Tx. Rest bytes of the PS_DIRCMD are ignored. 01 2 byte command : 2 bytes {DIRCMD_BYTE1,DIRCMD_BYTE0} of the PS_DIRCMD is transferred to the UART Tx without any idle time between DIRCMD_BYTE1 and DIRCMD_BYTE0. Rest byte of the PS_DIRCMD are ignored. 10/11 4 byte command : 4 bytes {DIRCMD_BYTE3,DIRCMD_BYTE2,DIRCMD_BYTE1,DIRCMD_BYTE0} of the PS_DIRCMD is transferred to the UART Tx line without any idle time between these bytes.

Table continues on the next page...

PSI5S_GLCR field descriptions (continued)

Field	Description
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 IE_DIRCMD_RDY	Interrupt enable bit for DIRCMD_RDY. **NOTE:** This bit is also writeable in PS_NORMAL mode. 0 PS_GLCR[DIRCMD_RDY], when goes as “1” , does not generateany interrupt on the “ipi_ps_glbl” interrupt line. 1 PS_GLCR[DIRCMD_RDY], when goes as “1” , generates an interrupt on the “ipi_ps_glbl” interrupt line.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 DEBUG_EN	Debug enable bit. For debug mode operation please refer Debug mode 0 The IP does not enter the debug mode when a request is made to the IP for entering into the debug mode. It continues its normal operation. 1 The IP enters the debug mode immediately, as soon as a request is made to the IP for entering into the debug mode.
14–15 GTM_TRIG_SEL	GTM_TRIG_SEL bits. These bits are used to select the trigger for the Timestamp counters as shown in Figure 57-21. The gtm trigger signals control the frequency of operation of the timestamp counters. In other words they control the clocking(clkA and clkB) of the timestamp counters A and B. 00 gtm_trig[0] is selected for clocking the timestamp counter. 01 gtm_trig[1] is selected for clocking the timestamp counter. 10 gtm_trig[2] is selected for clocking the timestamp counter. 11 gtm_trig[3] is selected for clocking the timestamp counter.
16 GL_DDSR_TRIG	This is the global shift trigger of all the DDSRs of each channel. When the corresponding PS_E2SCR_CH[n][GL_TRIG_SEL] == 1 , only then is this bit effective, else this bit has no effect. **NOTE:** If bit is subsequently set after being cleared , then the counters will restart and the offsets in PS_DDTRIG_PERR and PS_DDTRIG_OFFR registers will be applied again. This bit is also writable in PS_NORMAL mode 0 Shift triggers of all the DDSR registers are halted (default state). If bit is cleared during runtime, PS_DDTRIG_PERR and PS_DDTRIG_OFFR counters are reset. 1 The shift triggers of all the DDSR's start synchronously.
17 GL_MODETR_DONE_EN	Interrupt enable for PS_GLSR[GL_MODETR_DONE] bit. This bit is also writable in PS_NORMAL and PS_DISABLE modes. 0 No interrupt is generated on setting of the PS_GLSR[GL_MODETR_DONE] bit 1 Generates the interrupt when PS_GLSR[GL_MODETR_DONE] bit is set.
18 MRU_ERR_EN	Enables the interrupt for MRU overwrite when the unread contents of MRU Buffer1 are overwritten by a new message **NOTE:** This bit is also writable in ps_normal mode 0 MRU overwrite error is not enabled 1 MRU overwrite error is enabled
19 TSCS_B	This bit controls the selection of clock input to timestamp counter B, either by the external periodic clock or by the GTM module

Table continues on the next page...

PSI5S_GLCR field descriptions (continued)

Field	Description
	0 External clock(ipg_clk_ps_ts) is selected 1 gtm_trig is selected as the clock.
20 TSCS_A	This bit controls the selection of clock input to timestamp counter A, either by the external periodic clock or by the GTM module 0 External clock(ipg_clk_ps_ts) is selected 1 gtm_trig is selected as the clock.
21 CLR_CNTR_B	This bit will reset timestamp counter B when PS_GLCR[CLRTSCNT_G] is not set This bit is a one shot and is always read as 0. **NOTE:** This bit is also writable in ps_normal mode 1 Timestamp counter B is reset
22 CLR_CNTR_A	This bit will reset timestamp counter A when PS_GLCR[CLRTSCNT_G] is not set This bit is a one shot and is always read as 0. **NOTE:** This bit is also writable in ps_normal mode 1 Timestamp counter A is reset
23 CLRTSCNT_G	This bit controls the global clearing of the TimeStamp counters A and B when CLRTSCNT_G_L == 1. When CLRTSCNT_G_L == 0 then this bit has no effect. This bit is a one shot and is always read as 0. **NOTE:** This bit is also writable in ps_normal mode 1 Both time stamp counter are cleared at the same time.
24 CLRTSCNT_G_L	This bit control whether the timestamp counters counter A and B are cleared simultaneously or independently: 0 Timestamp counters A and B are cleared independently. When PS_GLCR[CLR_CNTR_A] == 1 timestamp counter A is cleared. When PS_GLCR[CLR_CNTR_A]== 1 timestamp counter B is cleared. 1 Both timestamp counter are cleared simultaneously as soon as CLRTSCNT_G== 1.
25 TSCNT_EN_B	This bit controls enabling of timestamp counter A when PS_GLCR[TSCNTEN_G] is not set **Note:** This bit is also writable in ps_normal mode 0 Timestamp counter A not enabled 1 Timestamp counter B is enabled
26 TSCNT_EN_A	This bit controls enabling of timestamp counter A when PS_GLCR[TSCNTEN_G] is not set **NOTE:** This bit is also writable in ps_normal mode 0 Timestamp counter A not enabled 1 Timestamp counter A is enabled
27 TSCNTEN_G	This bit controls the global trigger of the TimeStamp counters A and B when TSCNTEN_G_L == 1. When TSCNTEN_G_ == 0 then this bit has no effect. **NOTE:** This bit is also writable in ps_normal mode 0 Both time stamp counters are disabled simultaneously 1 Both time stamp counter start simultaneously.

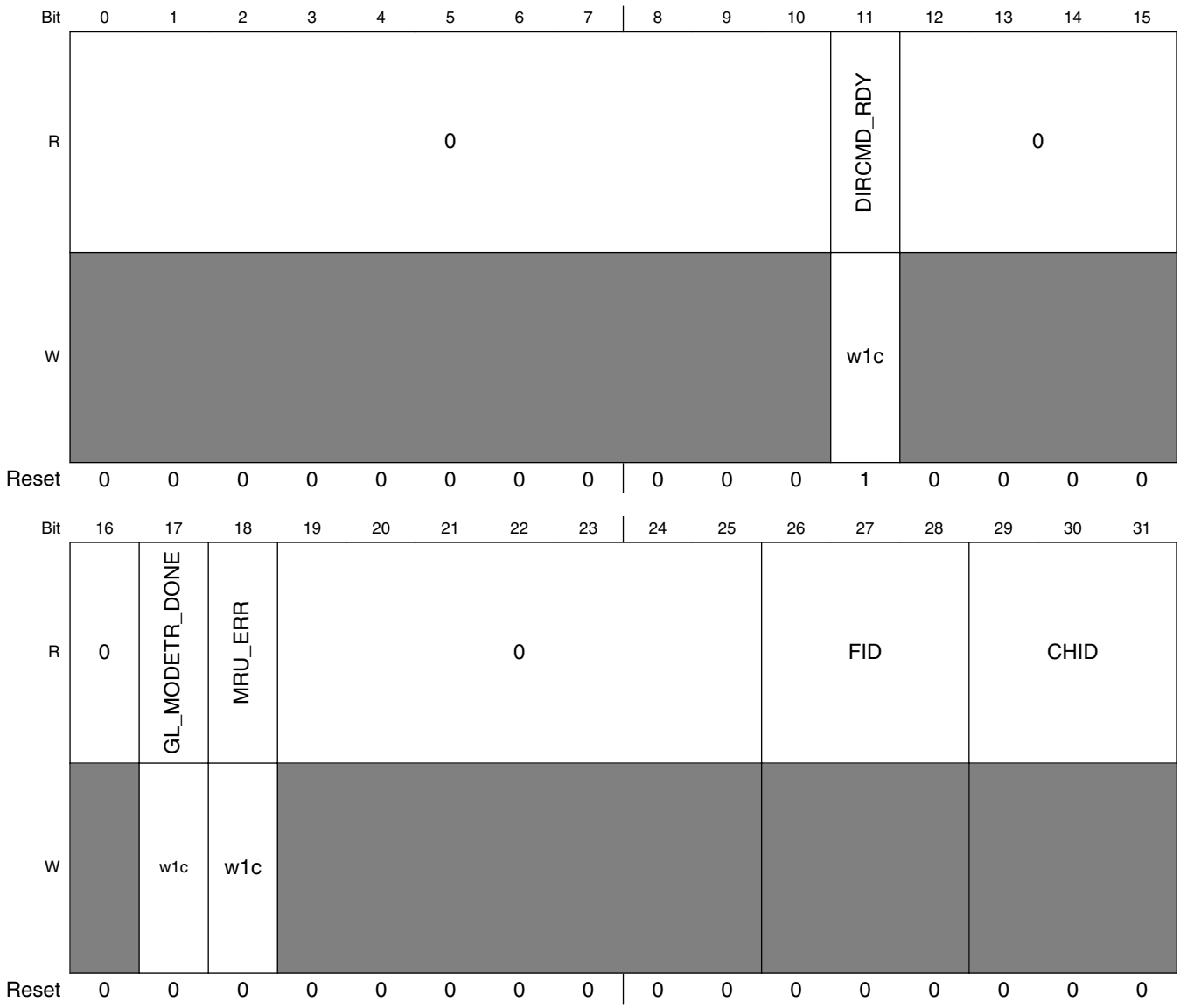
Table continues on the next page...

PSI5S_GLCR field descriptions (continued)

Field	Description
28 TSCNTEN_G_L	This bit control whether the timestamp counters counter A and B are started simultaneously or independently 0 Timestamp counters A and B are started independently. When PS_GLCR[TSCNT_A_EN] == 1 timestamp counter A starts. When PS_GLCR[TSCNT_B_EN] == 1 timestamp counter B starts. 1 Both timestamp counter are triggered simultaneously as soon as TSCNTEN_G == 1.
29–31 GLOBAL_MODE	These are the bits which decide Global Mode of the IP. There are four modes, UART_STDALONE, PS_DISABLE,PS_CONFIG and PS_NORMAL. For the details of modes related to the bit setting of GLOBAL_MODE, please refer Table 57-1. **NOTE:** This bits are writable in all modes

57.4.18 PSI5-S Global Status Register (PSI5S_GLSR)

Address: FBF7_4000h base + B8h offset = FBF7_40B8h



PSI5S_GLSR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 DIRCMD_RDY	Bit used to indicate the “Ready for Direct Command write” status.

Table continues on the next page...

PSI5S_GLSR field descriptions (continued)

Field	Description
	Once the command is written to the PS_DIRCMD, then a “w1c” to this register should be done so that the command written to the PS_DIRCMD register is queued for UART transmission with the highest priority. When the command has been transferred from thie register to the UART, then this bit goes as “1”. If the PS_GLCR[IE_DIRCMD_RDY] bit is set to “1”, then this event generates an interrupt on the ipi_ps_glbl line. **NOTE:** This bit is also writeable in the PS_NORMAL mode. 0 Direct Command that is written to the PS_DIRCMD register is currently under processing. Any further command written to the PS_DIRCMD register would be rejected without any information. 1 The PS_DIRCMD register is empty and the application can write the direct ECU to sensor transceiver command to the PS_DIRCMD register.
12–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 GL_MODETR_DONE	This bit indicates , as to when the global mode transitions , as programmed by PS_GLCR[GLOBAL_MODE] have actually been accomplished : This bit is cleared by w1c. The bit is also writable in PS_NORMAL and PS_DISABLE mode. This bit is reset to "0" when the mode is changed from PS_DISABLE to the UART_STDALONE mode. 0 The intended global mode , as programmed in the PS_GLCR[GLOBAL_MODE] register bits , is under processing. 1 The global mode as programmed in the PS_GLCR[GLOBAL_MODE] have been accomplished
18 MRU_ERR	This bit shows whether unread MRU buffer1 contents have been overwritten by a new message: This bit is cleared by w1c. The bit is also writable in PS_NORMAL mode. 0 no overwrite occurred 1 overwrite occurred
19–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–28 FID	Points to the frame whose data in MRU buffer 1 was overwritten
29–31 CHID	Points to the channel whose data in MRU buffer 1 was overwritten

57.4.19 PSI5-S CHANNEL_BASE_ADDRESS (PSI5S_CH_BASE_ADDR)

Address: FBF7_4000h base + BCh offset = FBF7_40BCh

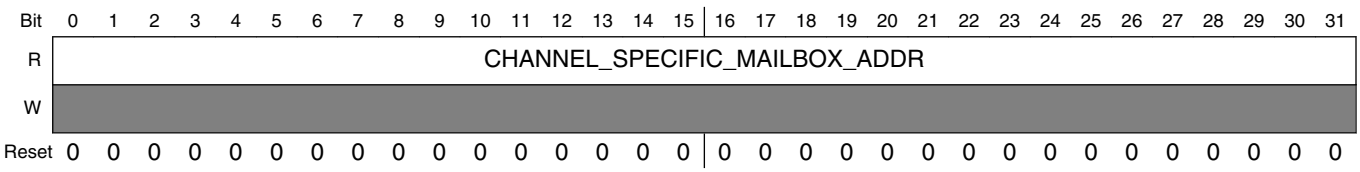
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MAILBOX_BASE_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_CH_BASE_ADDR field descriptions

Field	Description
0–31 MAILBOX_ BASE_ADDR	This register will provide the base address of the mailbox in the system RAM.This value needs to be programmed by the software.

57.4.20 PSI5-S MRU OUTPUT BUFFER2 REGISTER0 (PSI5S_MRU_BUF2_REG0)

Address: FBF7_4000h base + C0h offset = FBF7_40C0h



PSI5S_MRU_BUF2_REG0 field descriptions

Field	Description
0–31 CHANNEL_ SPECIFIC_ MAILBOX_ADDR	This register will provide the channel specific mailbox address corresponding to channel and slot for which data has been received.This value is computed by the IP based on the PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] and FID,CID. Following is the formula used to calculate this address: • For messages with CID==0 in the header byte, destined for Channel0 location0, the value is PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] • For "unrecoverable" messages destined for Channel0,location1, the value is PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR]+0xC • For other messages with CID = 1 to 7 the value can be calculated as (PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] + (6*4*3(MSG_CH_ID)+(MSG_FRAME_ID)*4*3)) When using the DMA for reading the MRU buffer, this register forms the Access1 of the DMA transfer.

57.4.21 PSI5-S MRU OUTPUT BUFFER2 REGISTER1 (PSI5S_MRU_BUF2_REG1)

Address: FBF7_4000h base + C4h offset = FBF7_40C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DCI				0	R_UVL_ERR	N_ ERR	CHID			FID			R_OVL_ERR	F_WD_ERR	SCI_O_ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SCI_F_ERR	SCI_P_ERR	HD_ERR	ERR		CRC_ERR_P_ERR	CRC			XCRC_ERR	XCRC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_MRU_BUF2_REG1 field descriptions

Field	Description
0–3 DCI	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 R_UVL_ERR	Message underflow. It is set when less than three UART bytes are received in the UART packet. In case of XCRC error the message is treated as unrecoverable and written in mail box location1 of channel 0 Please refer to PSI5-S Message Extraction for more details. 0 no error occurred 1 error occurred
6 N_ERR	Indicates that the number of UART bytes in the recieved UART packet does not match the number specified in Channel Configuration Register A indicated by CID and FID i.e. the number of UART bytes does not match the value programmed in PS_MSGA_CHCID[FFID_bytes]. Also please see the content about byte recalculation in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR). Please refer to PSI5-S Message Extraction for more details. 0 no error occurred 1 error occurred
7–9 CHID	Indicates Channel id of PSI5 message present in the UART header byte.
10–12 FID	Indicates Frame ID of PSI5 message present in the UART header byte.
13 R_OVL_ERR	Message overflow.It is set when more than six UART bytes are received in the UART packet.The message may be recoverable(XCRC_ERR == 0) or unrecoveranle (XCRC_ERR==1). Please refer to PSI5-S Message Extraction for more details. 0 no error occurred 1 error occurred
14 F_WD_ERR	Frame watchdog error in the received PSI5 message. In case there is no XCRC error in the recieved message, then this bit will be the same as the PS_WDGTSSR[F_WD_ERR[CID]] bit , where CID is the channel ID of the recieved PSI5 message. In case the recieved PSI5 message has an XCRC error, then this bit will be the logical OR of all the PS_WDGTSSR[F_WD_ERR[CHn]] bits, i.e., in this case, it will be “PS_WDGTSSR[F_WD_ERR[1]] OR PS_WDGTSSR[F_WD_ERR[2]] OR....PS_WDGTSSR[F_WD_ERR[7]]”. Please see PSI5-S Watchdog Operation, PSI5-S channel message configuration register B (PSI5S_MSGB_CH*n*) and PSI5-S Watchdog Error Status and Watchdog Timestamp status register (PSI5S_WDGTSSR) for more details. **NOTE:** Whenever the PS_MRU_BUF2_REG1[F_WD_ERR] bit is set, PS_WDGTSSR(which is a hardware clearing register) should always be read. Failing to do so will result in the PS_MRU_BUF2_REG1[F_WD_ERR] bit continuing to be set for respective “CID”, whenever the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit is set. This would happen in subsequent messages saved to the mailbox even though no further watchdog error has occurred for the respective CID.This happens since for each received PSI5 message in the MRU buffers the status of the corresponding PS_MRU_BUF2_REG1[F_WD_ERR] is picked from the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit. 0 no error occurred 1 error occurred

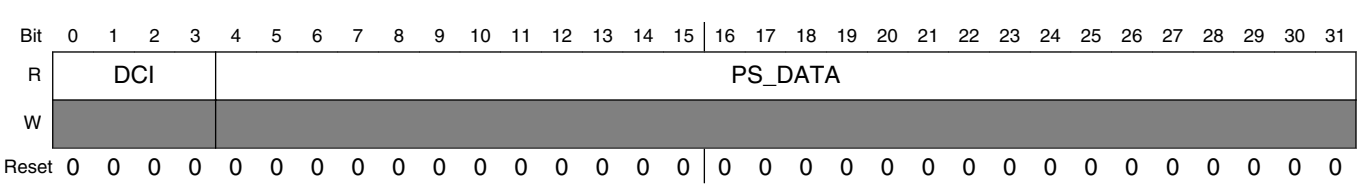
Table continues on the next page...

PSI5S_MRU_BUF2_REG1 field descriptions (continued)

Field	Description
15 SCI_O_ERR	UART overrun error has occured in at least one UART byte of the recieved UART packet , due to which some UART byte has been lost. 0 no error occurred 1 error occurred
16 SCI_F_ERR	UART framing error has occured in the current recieved UART packet. 0 no error occurred 1 error occurred
17 SCI_P_ERR	UART message parity error has occured in at least one UART byte of the recieved UART packet. 0 no error occurred 1 error occurred
18 HD_ERR	This bit is set when any of the ERR[1:0] is "1" and indicates a transceiver error. 0 no error occurred 1 error occurred
19–20 ERR	Transceiver error flags received in the Message header byte of the UART packet indicated by CID and FID
21 CRC_ERR_P_ERR	Indicates whether "PSI5 message " crc_err or the parity error in the PSI5 message indicated by CID and FID has occured . The Parity is taken as even for PSI5 caculation : **NOTE:** This bit always remains "0" for messages having CID and FID both as "0". 0 no crc_error/parity error has occurred 1 indicates crc_error/Parity Error has occurred
22–24 CRC	Contains the recieved CRC or Parity Bits of the PSI5 message, of indicated by CID and FID
25 XCRC_ERR	Indicates xcrc_err in the received UART packet : 0 no xcrc_error occurred 1 indicates xcrc_error occurred
26–31 XCRC	This is the value of xcrc received in the last six bits of the UART byte, which occurs just before the information of a "new UART packet recieved" has been generated. Please refer to PSI5-S UART Message Structure and PSI5-S XCRC Handling for more details.

57.4.22 PSI5-S MRU OUTPUT BUFFER2 REGISTER2 (PSI5S_MRU_BUF2_REG2)

Address: FBF7_4000h base + C8h offset = FBF7_40C8h



PSI5S_MRU_BUF2_REG2 field descriptions

Field	Description
0–3 DCI	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
4–31 PS_DATA	PSI5 data bits extracted from the recieved UART bytes.

57.4.23 PSI5-S MRU OUTPUT BUFFER2 REGISTER3 (PSI5S_MRU_BUF2_REG3)

Address: FBF7_4000h base + CCh offset = FBF7_40CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCI				0				TIMESTAMP																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

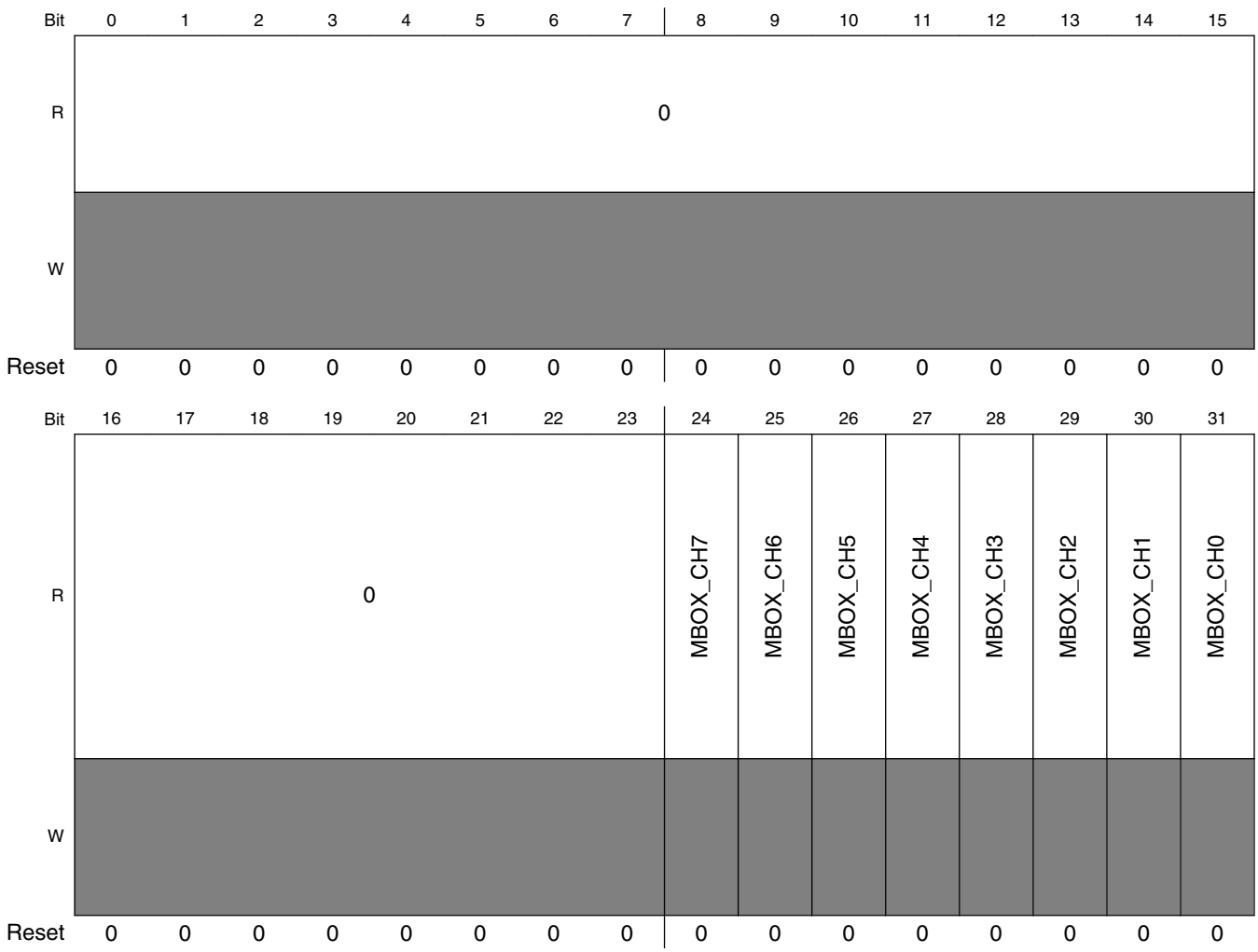
PSI5S_MRU_BUF2_REG3 field descriptions

Field	Description
0–3 DCI	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMESTAMP	Time stamp value appended based on the various timestamp bit settings in the PS_GLCR register,PS_PCCR_CH[n] register and the errors occurred. For more details please refer toPSI5-S Watchdog Operation.

57.4.24 PSI5-S Mbox Status Irq (PSI5S_MBOX_SR_IRQ)

Please refer to Figure 57-18 also.

Address: FBF7_4000h base + E0h offset = FBF7_40E0h



PSI5S_MBOX_SR_IRQ field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MBOX_CH7	When set it Indicates at least one message in mailbox 7 is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH7[F(0-5)_READ] bits i.e. MBOX_CH7 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel7 are cleared.

Table continues on the next page...

PSI5S_MBOX_SR_IRQ field descriptions (continued)

Field	Description
25 MBOX_CH6	When set it Indicates at least one message in mailbox 6 is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH6[F(0-5)_READ] bits i.e. MBOX_CH6 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel6 are cleared.
26 MBOX_CH5	When set it Indicates at least one message in mailbox 5 is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH5[F(0-5)_READ] bits i.e. MBOX_CH5 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel5 are cleared.
27 MBOX_CH4	When set it Indicates at least one message in mailbox 4 is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH4[F(0-5)_READ] bits i.e. MBOX_CH4 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel4 are cleared.
28 MBOX_CH3	When set it Indicates at least one message in mailbox 3 is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH3[F(0-5)_READ] bits i.e. MBOX_CH3 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel3 are cleared.
29 MBOX_CH2	When set it Indicates at least one message in mailbox 2 is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH2[F(0-5)_READ] bits i.e. MBOX_CH2 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel2 are cleared.
30 MBOX_CH1	When set it Indicates at least one message in mailbox 1is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH1[F(0-5)_READ] bits i.e. MBOX_CH1 = F0_READ OR F1_READ OR .....F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel1 are cleared.
31 MBOX_CH0	When set it Indicates at least one message in mailbox 0is pending for read by the application. This bit is a "read status" of the OR of PS_MBOX_SR_CH0[F(0-1)_READ] bits i.e. MBOX_CH0 = F0_READ OR F1_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel0 are cleared.

57.4.25 PSI5-S Error Status Irq (PSI5S_ERR_SR_IRQ)

Please refer to Figure 57-18 also.

Address: FBF7_4000h base + E4h offset = FBF7_40E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						R_UVL_ERR	N_ERR	R_OVL_ERR	F_WD_ERR	SCI_O_ERR	SCI_F_ERR	SCI_P_ERR	HD_ERR	CRC_ERR_P_ERR	XCRC_ERR
W							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_ERR_SR_IRQ field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 R_UVL_ERR	When set indicates that message underflow error(number of UART bytes recieved are less than 3) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[R_UVL_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[R_UVL_ERR] bit.

Table continues on the next page...

PSI5S_ERR_SR_IRQ field descriptions (continued)

Field	Description
	Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
23 N_ERR	When set indicates that N_ERR (Number of UART bytes are not equal to PS_MSGA_CHn[Fn_byte]) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[N_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[N_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
24 R_OVL_ERR	When set indicates that message Overflow error(number of UART bytes recieved have exceeded 6) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[R_OVL_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[R_OVL_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
25 F_WD_ERR	When set indicates that Frame Watchdog error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[F_WD_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[F_WD_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
26 SCI_O_ERR	When set indicates that UART Overrun error(from the UART side) is present in the message currently present in MRU buf2. When "1" , then this bit indicates that some UART byte belonging to the concerned UART packet , has been overwritten by another UART byte and is lost. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_O_ERR] bit.

Table continues on the next page...

PSI5S_ERR_SR_IRQ field descriptions (continued)

Field	Description
	This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_O_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
27 SCI_F_ERR	When set indicates that UART Framing error is present in the message currently present in MRU buf2. If this bit is set then it indicates to the application that the synchronisation of the UART module is lost and any subsequent UART bytes recieved will be error prone. In this case the application is required to reset the UART module by using the PS_GCR[SR] bit. Subsequently it should restart the transmission. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_F_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_F_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
28 SCI_P_ERR	When set indicates that UART parity error is present in the message currently present in MRU buf2. This bit will be set when there is a parity error in any UART byte using which the concerned UART packet was formed. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_P_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_P_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
29 HD_ERR	When set indicates that Header error(E0/E1) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[HD_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[HD_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.

Table continues on the next page...

PSI5S_ERR_SR_IRQ field descriptions (continued)

Field	Description
30 CRC_ERR_P_ERR	When set indicates that PSI5 CRC/Parity error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[CRC_ERR/P_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[CRC_ERR/P_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** This bit always remains "0" for messages having CID and FID both as "0". **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.
31 XCRC_ERR	When set indicates that XCRC error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[XCRC_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[XCRC_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **NOTE:** If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message.

57.4.26 PSI5-S Mailbox select IRQ[irq_n] (PSI5S_MBOX_SEL_IRQ*n*)

Please refer to Figure 57-18 also.

Address: FBF7_4000h base + E8h offset + (8d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								MBOX_CH7_SEL	MBOX_CH6_SEL	MBOX_CH5_SEL	MBOX_CH4_SEL	MBOX_CH3_SEL	MBOX_CH2_SEL	MBOX_CH1_SEL	MBOX_CH0_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_MBOX_SEL_IRQ*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MBOX_CH7_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH7] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH7] contributes to generation of irq[irq_n] interrupt.
25 MBOX_CH6_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH6] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH6] contributes to generation of irq[irq_n] interrupt.
26 MBOX_CH5_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH5] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH5] contributes to generation of irq[irq_n] interrupt.
27 MBOX_CH4_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH4] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH4] contributes to generation of irq[irq_n] interrupt.

Table continues on the next page...

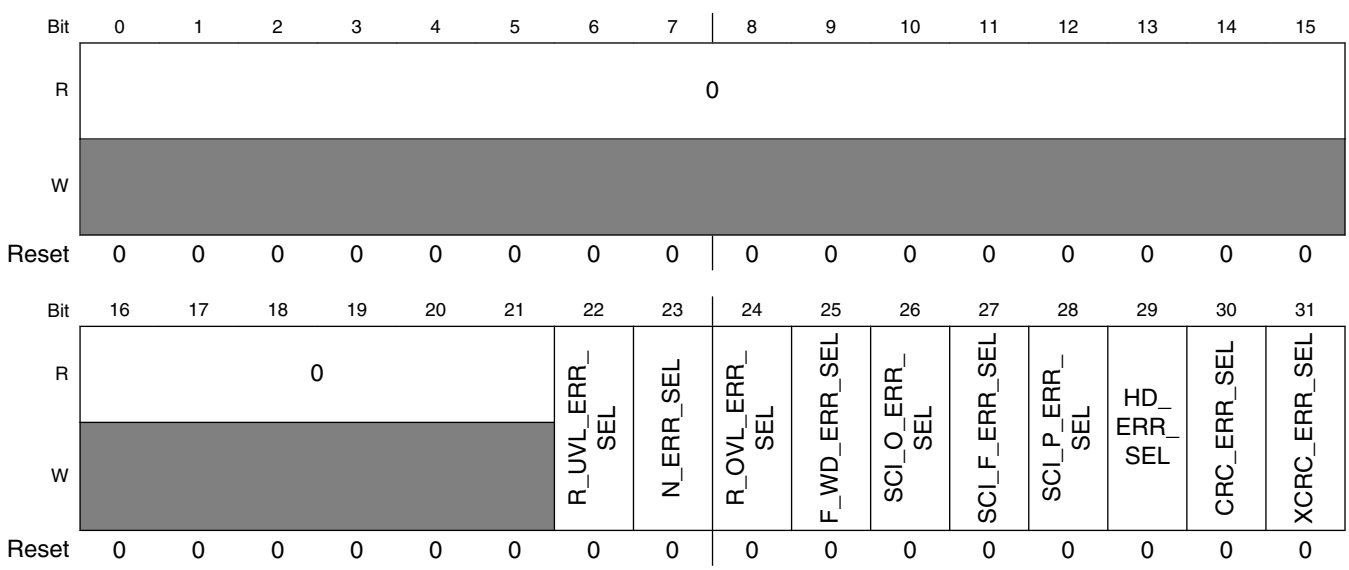
PSI5S_MBOX_SEL_IRQ*n* field descriptions (continued)

Field	Description
28 MBOX_CH3_ SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH3] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH3] contributes to generation of irq[irq_n] interrupt.
29 MBOX_CH2_ SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH2] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH2] contributes to generation of irq[irq_n] interrupt.
30 MBOX_CH1_ SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH1] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH1] contributes to generation of irq[irq_n] interrupt.
31 MBOX_CH0_ SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_MBOX_SR_IRQ[MBOX_CH0] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH0] contributes to generation of irq[irq_n] interrupt.

57.4.27 PSI5-S Error Select IRQ[iq_n] (PSI5S_ERR_SEL_IRQ*n*)

Please refer to Figure 57-18 also.

Address: FBF7_4000h base + ECh offset + (8d × i), where i=0d to 7d



PSI5S_ERR_SEL_IRQ*n* field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 R_UVL_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[R_UVL_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[R_UVL_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
23 N_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[N_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[N_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
24 R_OVL_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[R_OVL_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[R_OVL_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
25 F_WD_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[F_WD_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[F_WD_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
26 SCI_O_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[SCI_O_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_O_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
27 SCI_F_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[SCI_F_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_F_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
28 SCI_P_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[SCI_P_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_P_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
29 HD_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[HD_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[HD_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
30 CRC_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode

Table continues on the next page...

PSI5S_ERR_SEL_IRQ*n* field descriptions (continued)

Field	Description
	0 PS_ERR_SR_IRQ[CRC_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[CRC_ERR_SEL] contributes to generation of irq[irq_n] interrupt.
31 XCRC_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt: It is also writable in ps_normal mode 0 PS_ERR_SR_IRQ[XCRC_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[XCRC_ERR_SEL] contributes to generation of irq[irq_n] interrupt.

57.4.28 PSI5-S Watchdog Error Status and Watchdog Timestamp status register (PSI5S_WDGTSSR)

NOTE

All bits of PS_WDGTSSR register are self clearing on read. This implies that in case this register is read by the application before the corresponding PS_WDGTSSR[F_WD_ERR_STATUS[CHn]] has had a chance to be sent in the next upcoming PSI5 message, then this bit would be sent as a “0” in this upcoming PSI5 message.

Address: FBF7_4000h base + 128h offset = FBF7_4128h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_WD_ERR_STATUS							0	WDTS_STATUS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDTS_STATUS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_WDGTSSR field descriptions

Field	Description
0–6 F_WD_ERR_ STATUS	These bits give the watchdog error status for the watchdog of each corresponding channel. Thus, F_WD_ERR_STATUS[1] will give the watchdog error status for watchdog of channel1. Similarly, F_WD_ERR_STATUS[2] will give the watchdog error status for watchdog of channel2 and so on till F_WD_ERR_STATUS[7], which gives the watchdog error status for watchdog of channel7. As soon as the watchdog of the corresponding channel has a watchdog overrun error, the error status is latched in the corresponding bits of this register. **NOTE:** PS_WDGTSSR is a hardware cleared register. These bits are automatically cleared by the hardware when PS_WDGTSSR is read by the application. This register should ALWAYS be read

Table continues on the next page...

PSI5S_WDGTSSR field descriptions (continued)

Field	Description
	whenever the PS_MRU_BUF2_REG1[F_WD_ERR] bit is set. Failing to do so will result in the PS_MRU_BUF2_REG1[F_WD_ERR] bit continuing to be set for respective “CID” whenever the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit is set. This would happen in subsequent messages saved to the mailbox even though no further watchdog error has occurred for the respective CID. This happens since for each received PSI5 message in the MRU buffers the status of the corresponding PS_MRU_BUF2_REG1[F_WD_ERR] is picked from the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 WDTS_STATUS	Timestamp value of that specific channel , the timestamp of which is captured when the watchdog error of that specific channel has occured. This register always contains the timestamp value corresponding to the last occuring Watch Dog error. If due to bad programming watchdog error of more than one channel happen simultaneously then this register contains the timestamp value of the highest channel. For each channel , whether Time Stamp A or B is captured depends on the value of PS_MSGA_CHn[TIMESTAMP_A_B_SEL] register bit. **NOTE:** These bits are automatically cleared by the hardware when any of the bits of PS_WDGTSSR is read by the application.

57.4.29 PSI5-S ECU to Sensor Direct Command Write register (PSI5S_DIRCMD)

NOTE

A “w1c” should be done on the PS_GLSR[DIRCMD_RDY] so that the command written on the PS_DIRCMD register is queued for transmission on the UART TX line. As soon as any ongoing transmission on the UART TX is finished, the PS_DIRCMD starts being transmitted on the UART Tx line. When PS_GCR[TDFBM] = 0, then the bits on the UART Tx line are shifted in the following sequence “DIRCMD_BYTE0[0],DIRCMD_BYTE0[1]....DIRCMD_BYTE3[0]....DIRCMD_BYTE3[7]” with DIRCMD_BYTE0[0] being shifted first.

Address: FBF7_4000h base + 12Ch offset = FBF7_412Ch

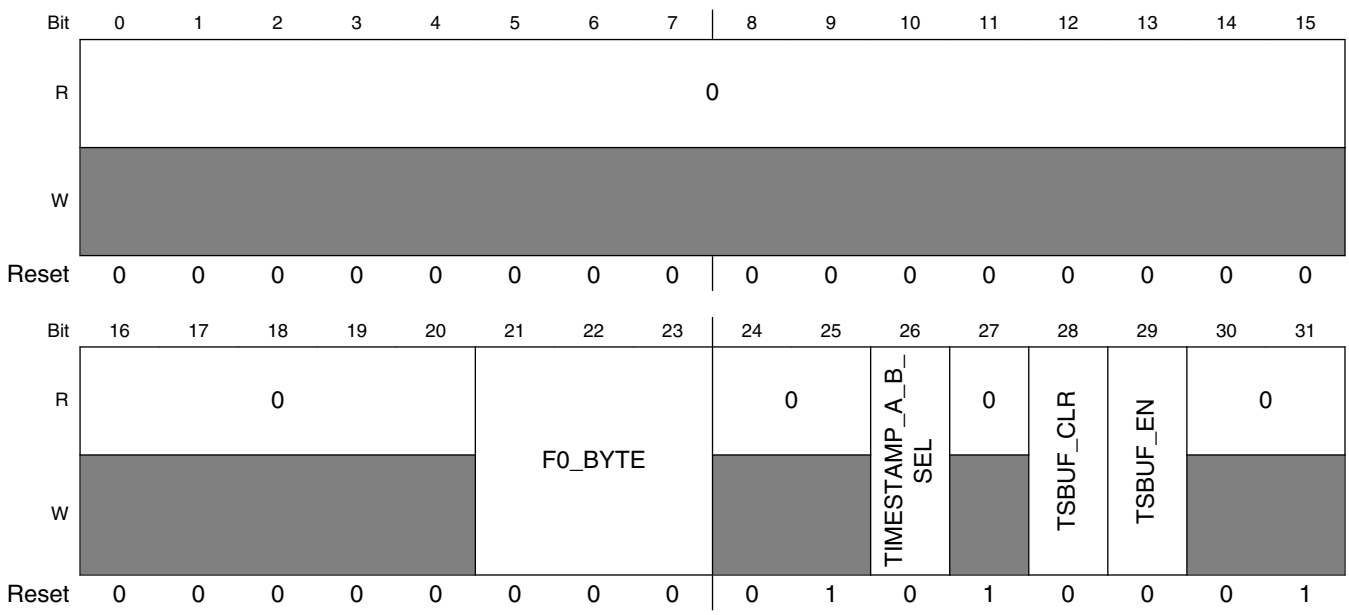
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIRCMD_BYTE3[7:0]								DIRCMD_BYTE2[7:0]								DIRCMD_BYTE1[7:0]								DIRCMD_BYTE0[7:0]							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_DIRCMD field descriptions

Field	Description
0–7 DIRCMD_ BYTE3[7:0]	Byte3 of the Direct Command. The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART. **NOTE:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is “1”.
8–15 DIRCMD_ BYTE2[7:0]	Byte2 of the Direct Command. The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART. **NOTE:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is “1”.
16–23 DIRCMD_ BYTE1[7:0]	Byte1 of the Direct Command. The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART. **NOTE:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is “1”.
24–31 DIRCMD_ BYTE0[7:0]	Byte0 of the Direct Command. The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART. **NOTE:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is “1”.

57.4.30 PSI5-S channel 0 message configuration register A (PSI5S_MSGA_CH0)

Address: FBF7_4000h base + 16Ch offset = FBF7_416Ch



PSI5S_MSGA_CH0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 F0_BYTE	Number of UART bytes comprising one UART packet. Applicable only to special messages having CID = 0 in their Message Header. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled and a message with the corresponding frame ID is not expected to occur. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TIMESTAMP_A_B_SEL	Whether the Timestamp corresponding to TimeStamp Counter A or B, is appended to messages targetted for Channel0: 0 Timestamp counter A 1 Timestamp counter B
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TSBUF_CLR	This bit is used to clear the local timestamp buffer for the respective channel. Once cleared, the buffer remains clear till the event to capture time stamp is triggered. This bit is one shot and is always read as "0". **NOTE:** This bit is also writeable in PS_NORMAL mode. 1 TimeStamp buffer is cleared.
29 TSBUF_EN	This bit controls whether timestamp buffer maintains a value "0" or is refreshed on a timestamp "capture event" 0 Timestamp buffer maintains value "0" 1 Timestamp buffer responds to a "capture event" and captures the respective timestamp value.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.31 PSI5-S channel 0 message configuration register B (PSI5S_MSGB_CH0)

Address: FBF7_4000h base + 170h offset = FBF7_4170h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																F0_payload															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

PSI5S_MSGB_CH0 field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 F0_payload	Number of bits in payload region of the message (only for channels having CID as "0" in the message header). This channel is always asynchronous. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

57.4.32 PSI5-S Mailbox status register channel0 (PSI5S_MBOX_SR_CH0)

Please refer to Figure 57-18 also.

Address: FBF7_4000h base + 178h offset = FBF7_4178h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										F1_ ERR	F1_ OV	F1_READ	F0_ ERR	F0_ OV	F0_READ
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_MBOX_SR_CH0 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 F1_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when an unrecoverable message(with the CID = 0 and FID=1) is read from the MRU_BUF2. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC1. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC1) by the DMA , has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC1) by the DMA , has one or more _ERR bits set.
27 F1_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC1. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX0,LOC1. 1 Indicates that there is an unread message stored in MBOX0,LOC1 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
28 F1_READ	MBOX0,LOC1 is reserved for any unrecoverable message(XCRC_ERR has occurred or it is an "illegal message"). It can correspond to any channel from 1 to 7. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC1. This bit is also writeable in ps_normal mode. 0 No unread message in MBOX0,LOC1. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX0,LOC1 by the DMA.
29 F0_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when a transceiver message(with the CID = 0 and FID=0) is read from the MRU_BUF2. It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC0. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC0) by the DMA , has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0,LOC0) by the DMA , has one or more _ERR bits set.
30 F0_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC0. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX0,LOC0. 1 Indicates that there is an unread message stored in MBOX0,LOC0 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
31 F0_READ	MBOX0,LOC0 is reserved for a transceiver message . It is a w1c and software should clear it whenever it reads the message from the MBOX0,LOC0. This bit is also writeable in ps_normal mode.

Table continues on the next page...

PSI5S_MBOX_SR_CH0 field descriptions (continued)

Field	Description
	0 No unread message in MBOX0,LOC0. 1 It indicates that the message that has just been read from PS_MRU_BUF2 will be stored in MBOX0,LOC0 by the DMA.

57.4.33 PSI5-S channel message configuration register A (PSI5S_MSGA_CH*n*)

Address: FBF7_4000h base + 190h offset + (60d × i), where i=0d to 6d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L_PC_EN	F5_BYTE			L_PC5	F4_BYTE			L_PC4	F3_BYTE			L_PC3	F2_BYTE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L_PC2	F1_BYTE			L_PC1	F0_BYTE			L_PC0	MODE	TIME_STAMP_A_B_SEL	TMSG_TCMD	TSBUF_CLR	TSBUF_EN	G_PC	CH_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_MSGA_CH*n* field descriptions

Field	Description
0 L_PC_EN	Selects whether the parity/CRC selection for a PSI5 message is to be controlled globally on a channel basis (through the G_PC bits) or locally on a per frame basis (through the L_PC[frame] bits) 0 Parity/CRC option controlled globally through the G_PC bit 1 Parity/CRC option controlled locally through the L_PC[frame] bits individually for message of each frame.
1–3 F5_BYTE	Number of bytes in UART packet for message corresponding to frame 5. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F5_payload] is immaterial. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
4 L_PC5	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message

Table continues on the next page...

PSI5S_MSGA_CH*n* field descriptions (continued)

Field	Description
	0 Parity selected for message for frame5 1 CRC selected for PSI5 message for frame5.
5–7 F4_BYTE	Number of bytes in UART packet for message corresponding to frame 4. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F4_payload] is immaterial. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
8 L_PC4	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame4 1 CRC selected for PSI5 message for frame4
9–11 F3_BYTE	Number of bytes in UART packet for message corresponding to frame 3. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F3_payload] is immaterial. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
12 L_PC3	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame3 1 CRC selected for PSI5 message for frame3.
13–15 F2_BYTE	Number of bytes in UART packet for message corresponding to frame 2. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F2_payload] is immaterial. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
16 L_PC2	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame2 1 CRC selected for PSI5 message for frame2.
17–19 F1_BYTE	Number of bytes in UART packet for message corresponding to frame 1. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F1_payload] is immaterial. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
20 L_PC1	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame1 1 CRC selected for PSI5 message for frame1

Table continues on the next page...

PSI5S_MSGA_CH*n* field descriptions (continued)

Field	Description
21–23 F0_BYTE	Number of bytes in UART packet for message corresponding to frame 0. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled(unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F0_payload] is immaterial. Please see the information in PSI5-S LIN Fractional Baud Rate Register (PSI5S_LINFBRR) that explains the recalculation of Fn_bytes.
24 L_PC0	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame0 1 CRC selected for PSI5 message for frame0.
25 MODE	This bit selects the operating modes between sync and async for the channel. 0 Synchronous 1 Asynchronous
26 TIME_STAMP_A_B_SEL	Decides whether Timestamp Counter A or B, value is captured in the local timestamp buffer, whenever a "capture event" occurs: 0 Timestamp counter A 1 Timestamp counter B.
27 TMSG_TCMD	Whether the timestamp is recorded when the message is received or when the channel command is moved from the TxFIFO to the UART Tx register (emulates a sync pulse) 0 Timestamp recorded based on "emulated" sync pulse. 1 Timestamp recorded based on message header byte received
28 TSBUF_CLR	This bit is used to clear the local timestamp buffer for the respective channel. Once cleared, the buffer remains clear till the event to capture time stamp is triggered. This bit is one shot and is always read as "0". **NOTE:** This bit is also writeable in PS_NORMAL mode. 1 TimeStamp buffer is cleared.
29 TSBUF_EN	This bit controls whether timestamp buffer maintains a value "0" or is refreshed on a timestamp "capture event" 0 Timestamp buffer maintains value "0" 1 Timestamp buffer responds to a "capture event" and captures the respective timestamp value.
30 G_PC	This bit globally selects the Parity or the CRC option for all the Frames of that particular channel provided L_PC_EN is set "0" 0 Parity option selected globally for messages of all frames. 1 CRC option selected globally for messages of all frames.
31 CH_EN	PSI5 Channel Enable. This bit decides if the IP takes the data belonging to the specific channel as "legal" or "illegal". 0 Concerned PSI5 channel is disabled.Any data occurring with a CID value of a disabled channel is treated as an "illegal message" and send to Channel 0 unrecoverable message area(MBOX0,LOC1) 1 Concerned PSI5 channel is enabled.Any data occurring with a CID value of an enabled channel can either be treated as "legal" or "illegal". Please see PSI5-S Message extraction for the definition of legal

Table continues on the next page...

PSI5S_MSGA_CH*n* field descriptions (continued)

Field	Description
	and illegal messages. Legal messages are sent to MBOXCID,LOCFID while illegal messages are sent to MBOX0,LOC1

57.4.34 PSI5-S channel message configuration register B (PSI5S_MSGB_CH*n*)

NOTE

Fn_byte recalculation

The IP expects legal values to be programmed for Fn_byte field. If this is not followed then the IP reprogrammes the registers with default values. Whenever the value of “Fn_byte” field becomes 1 or 2 it is defaulted to 3 and value of 7 is defaulted to 6. This updation happens once the IP enters the PS_NORMAL mode.

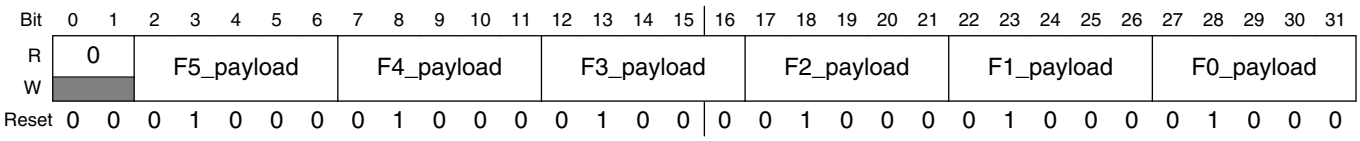
In addition to the above , the following checks are also performed.

By default the IP will expect that there is coherence between the expected number of bytes calculated through the “minUARTbytes” and the number of bytes in the “Fn_byte” field. Please see Figure 57-5 for definition of “minUARTbytes”. If there is no coherence (i.e. the above two calculations do not match) then the IP does the following recalculation:

- When Programmed Fn_byte > “minUARTbytes” , then the IP will expect that the additional extra bytes would be stuffed with 0s from the transceiver. Rest of the checking will be done as described in PSI5-S "Recoverable and Unrecoverable" Messages Structure.
- When Fn_byte < “minUARTbytes”, then the “Fn_bytes” is internally “increased” such that the expanded “Fn_bytes” is now equal to minUARTbytes”. Please see Figure 57-5 for definition of “minUARTbytes”. Using this expanded “Fn_bytes” value the message that would be arriving is analyzed. This would mean that no data bits are lost. Rest of the checking will be done as described in PSI5-S "Recoverable and Unrecoverable" Messages Structure.

The “recaculated” Fn_byte field is automatically updated by the IP once the IP enters the NORMAL mode of operation. For the subsequent occurence in the document , all references to “Fn_byte” would mean the recalculated value of “Fn_byte”.

Address: FBF7_4000h base + 194h offset + (60d × i), where i=0d to 6d



PSI5S_MSGB_CH*n* field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–6 F5_payload	Number of bits in the Payload region of the PSI5 message corresponding to frame 5. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
7–11 F4_payload	Number of bits in the Payload region of the PSI5 message corresponding to frame 4. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
12–16 F3_payload	Number of bits in the Payload region of the PSI5 message corresponding to frame 3. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
17–21 F2_payload	Number of bits in the Payload region of the PSI5 message corresponding to frame 2. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
22–26 F1_payload	Number of bits in the Payload region of the PSI5 message corresponding to frame 1. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
27–31 F0_payload	Number of bits in the Payload region of the PSI5 message corresponding to frame 0. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

57.4.35 PSI5-S Mailbox status register channel (PSI5S_MBOX_SR_CH*n*)

Please refer to Figure 57-18 also.

Address: FBF7_4000h base + 19Ch offset + (60d × i), where i=0d to 6d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														F5_ ERR	F5_ OV
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F5_READ	F4_ ERR	F4_ OV	F4_READ	F3_ ERR	F3_ OV	F3_READ	F2_ ERR	F2_ OV	F2_READ	F1_ ERR	F1_ OV	F1_READ	F0_ ERR	F0_ OV	F0_READ
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_MBOX_SR_CH*n* field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 F5_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message(with the CID = n and FID=5) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC5. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC5) by the DMA, has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC5) by the DMA, has one or more _ERR bits set.
15 F5_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC5. This bit is also writeable in ps_normal mode.

Table continues on the next page...

PSI5S_MBOX_SR_CH*n* field descriptions (continued)

Field	Description
	0 No overwrite of unread message in MBOX[n],LOC5. 1 Indicates that there is an unread message stored in MBOX[n],LOC5 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
16 F5_READ	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC5. This bit is also writeable in ps_normal mode. 0 No unread message in MBOX[n],LOC5. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC5.
17 F4_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ registe. It is updated when the corresponding message(with the CID = n and FID=4) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC4. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC4) by the DMA , has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC4) by the DMA , has one or more _ERR bits set.
18 F4_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC4. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX[n],LOC4. 1 Indicates that there is an unread message stored in MBOX[n],LOC4 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
19 F4_READ	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC4. This bit is also writeable in ps_normal mode. 0 No unread message in MBOX[n],LOC4. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC4.
20 F3_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message(with the CID = n and FID=3) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC3. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC3) by the DMA, has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC3) by the DMA, has one or more _ERR bits set.
21 F3_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC3. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX[n],LOC3. 1 Indicates that there is an unread message stored in MBOX[n],LOC3 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.

Table continues on the next page...

PSI5S_MBOX_SR_CH*n* field descriptions (continued)

Field	Description
22 F3_READ	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC3. This bit is also writeable in ps_normal mode. 0 No unread message in MBOX[n],LOC3. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC3 by the DMA.
23 F2_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message(with the CID = n and FID=2) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC2. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC2) by the DMA, has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC2) by the DMA , has one or more _ERR bits set.
24 F2_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC2. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX[n],LOC2. 1 Indicates that there is an unread message stored in MBOX[n],LOC2 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
25 F2_READ	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC2. This bit is also writeable in ps_normal mode. 0 No unread message in MBOX[n],LOC2. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC2.
26 F1_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message(with the CID = n and FID=1) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC1. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC1) by the DMA, has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC1) by the DMA, has one or more _ERR bits set.
27 F1_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC1. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX[n],LOC1. 1 Indicates that there is an unread message stored in MBOX[n],LOC1 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
28 F1_READ	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC1. This bit is also writeable in ps_normal mode.

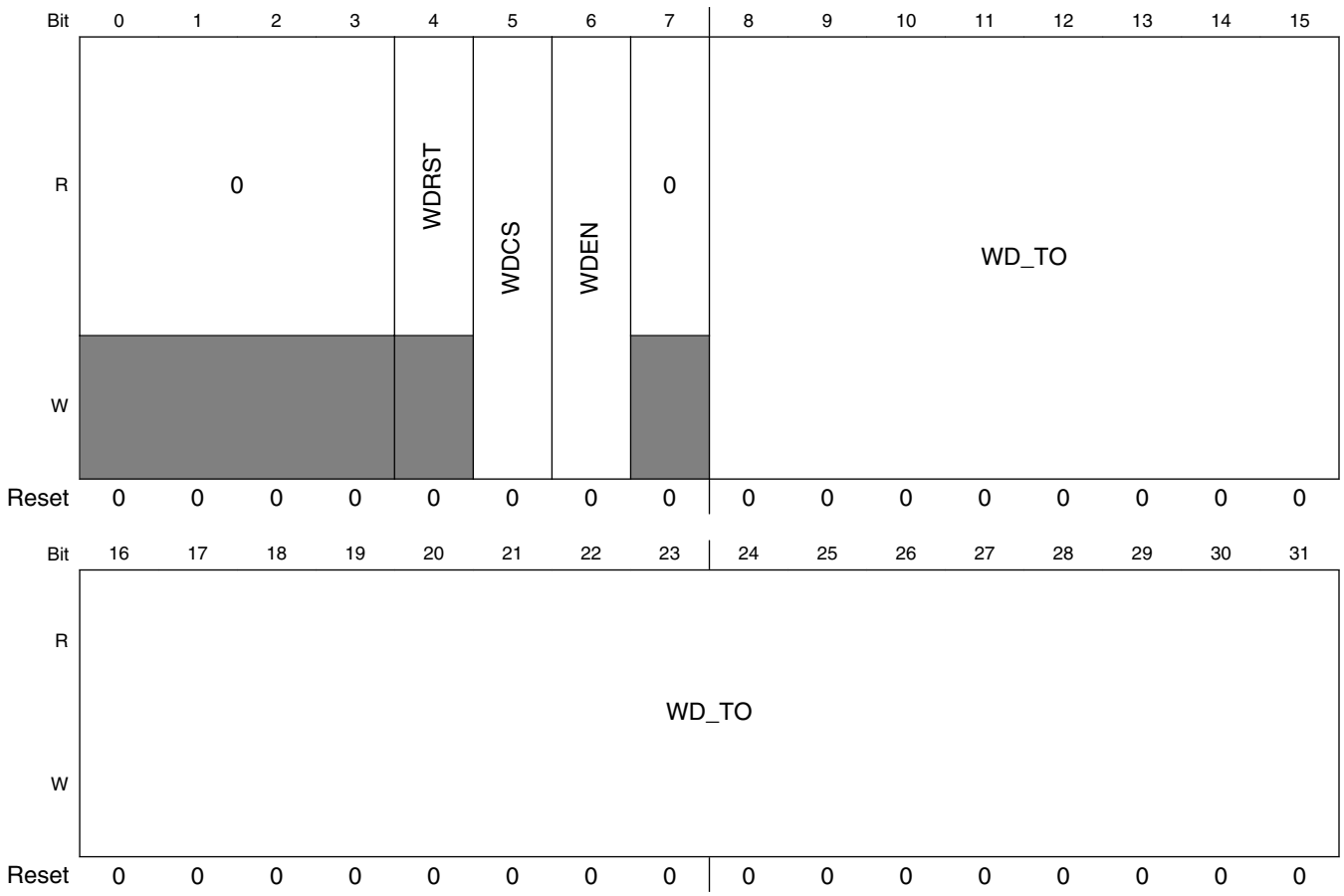
Table continues on the next page...

PSI5S_MBOX_SR_CH*n* field descriptions (continued)

Field	Description
	0 No unread message in MBOX[n],LOC1. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC1.
29 F0_ERR	This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message(with the CID = n and FID=0) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC0. This bit is also writeable in ps_normal mode. Please refer to Figure 57-18 also. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC0) by the DMA , has ALL the "_ERR" bits as "0". 1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n],LOC0) by the DMA , has one or more _ERR bits set.
30 F0_OV	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC0. This bit is also writeable in ps_normal mode. 0 No overwrite of unread message in MBOX[n],LOC0. 1 Indicates that there is an unread message stored in MBOX[n],LOC0 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA.
31 F0_READ	It is a w1c and software should clear it whenever it reads the message from the MBOX[n],LOC0. This bit is also writeable in ps_normal mode. 0 No unread message in MBOX[n],LOC0. 1 It indicates that the message that has just been read from PS_MRU_BUF2 , will be stored in MBOX[n],LOC0.

57.4.36 PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH*n*)

Address: FBF7_4000h base + 1A0h offset + (60d × i), where i=0d to 6d



PSI5S_WD_CFGR_CH*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WDRST	Resets watchdog counter for the channel This bit is one shot and is also writable in ps_normal mode. It is always read as 0. It will reset the watchdog counter and the counter will be reloaded with the value of WD_TO and automatically start counting again.
5 WDCS	This will select either GTM or module clock for watchdog counter of the channel. Please see Figure 57-21. 0 ipg_clk_ps_wd Clock 1 gtm_trig is selected as the clock.
6 WDEN	The watchdog counter for each channel can be started, stopped. This bit is also writable in ps_normal mode

Table continues on the next page...

PSI5S_WD_CFGR_CH*n* field descriptions (continued)

Field	Description
	0 watchdog counter stopped. It maintains its value. 1 watchdog counter started
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 WD_TO	This indicates the watchdog time out value. The watchdog generates an overrun error, in case the watchdog is not refreshed before the watchdog counter reached the WD_TO value. The Watchdog is refreshed in a different manner depending on whether the PSI5-S channel is in asynchronous mode or synchronous mode as described below: In Asynchronous mode: As soon as the IP enters the PS_NORMAL mode, the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value, and in case the watchdog is enabled, then the counter starts immediately. Whether the WD has to be "refreshed" or "not to be refreshed", is analyzed when the XCRC is calculated. In case the XCRC is correct, the watchdog is refreshed (It is not STOPPED), else the WD counter keeps on running normally eventually leading to a watchdog overrun error. In case of watchdog overrun error, the counter is refreshed. In Synchronous Mode: As soon as the IP enters the PS_NORMAL mode, the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value and in case the watchdog is enabled, then the counter starts immediately. Thereby, it is re-loaded, either on the occurence of a sync pulse or the watchdog overrun error. In case the event for this load is the Sync Pulse, then it immediately starts counting again and keeps on counting till it is stopped when the “Packet_counter” reaches the count of “total number of configured slots”. The “Packet_counter” here is a counter that keeps track of the number of configured(enabled) slots. It is incremented on the reception of each UART packet with a “CID” and “FID” such that PS_MSGA_CHCID[FFID_byte] is not equal to “0”. The packet counter is, however, incremented ONLY if the XCRC of the received UART packet is correct else it is not incremented. In case the “Packet Counter” does not reach the “expected packet count” within the “WD timeout configured“, the WD overrun error is generated. In case of a WD overrun error, the counter is stopped and does not start till a Sync Pulse is encountered. In both the above modes, the watchdog can always be refreshed using a software reload. **NOTE:** The watchdog counter is enabled ONLY if the channel is enabled.

57.4.37 PSI5-S DDSR Trigger offset register channel (PSI5S_DDTRIG_OFFR_CH*n*)

Address: FBF7_4000h base + 1A4h offset + (60d × i), where i=0d to 6d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																DDTRIG_OFFR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_DDTRIG_OFFR_CH*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5S_DDTRIG_OFFR_CH*n* field descriptions (continued)

Field	Description
16–31 DDTRIG_OFFR	These bits configure the delay between the "shift trigger" of the various DDSRs across different channels. The value in this register governs the delay after which the specific DDSR would be triggered once the Global Shift trigger (PS_GLCR[GL_DDSR_TRIG]) is written to "1" and the corresponding PS_E2SCR_CH[n][GL_TRIG_SEL] is set to "1" for that channel. There is an internal counter which is clocked on the "ipg_clk_ps_ddtrig" clock. The counters count up to the value as programmed in the PS_DDTRIG_PERR_CH[n] registers after which they are reset. At each reset a shift trigger is generated for the DDSR. The "start" of these counters is delayed based on the value programmed in the PS_DDTRIG_OFFR_CH[n] register

57.4.38 PSI5-S DDSR Trigger period register channel (PSI5S_DDTRIG_PERR_CH*n*)

Address: FBF7_4000h base + 1A8h offset + (60d × i), where i=0d to 6d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																DDTRIG_PERR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_DDTRIG_PERR_CH*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 DDTRIG_PERR	These bits configure the "shift trigger" period for the DDSR of that specific channel. The value in this register governs the period with which the data for the DDSR of that particular channel would be shifted out. There is an internal counter which is clocked on the "ipg_clk_ps_ddtrig" clock. The counters count up to the value as programmed in the PS_DDTRIG_PERR_CH[n] registers after which they are reset. At each reset, a shift trigger is generated for the DDSR. The "start" of these counters is delayed based on the value programmed in the PS_DDTRIG_OFFR_CH[n] register . **NOTE:** This register should be programmed to be greater than or equal to 2.

57.4.39 PSI5-S ECU to Sensor Control Register (PSI5S_E2SCR_CH*n*)

The following table corresponds to the CMD_TYPE field of this register.

Table 57-5. CMD_TYPE meanings

CMD_TYPE	PSI5 Frame	Number of bits writable by the Application in the DDSR
"000" - 0	Short Frame with 31 "1s" as the start condition[1]	6 LSBs of DDSR_L
"001" - 1	Short Frame with 5 "0s" as the start condition[1]	6 LSBs of DDSR_L
"010" - 2	Long Frame with 31 "1s" as the start condition[1]	16 LSBs of DDSR_L
"011" - 3	Long Frame with 5 "0s" as the start condition[1]	16 LSBs of DDSR_L
"100" - 4	X-Long Frame with 31 "1s" as the start condition[1]	22 LSBs of DDSR_L
"101" - 5	X-Long Frame with 5 "0s" as the start condition[1].	22 LSBs of DDSR_L
"110" - 6	XX-Long	24 LSBs of DDSR_L
"111" - 7	Illegal	

1. The start condition (31 “1s” or 5 “0s”) is automatically taken care by the hardware and should NOT be written to the registers as part of the command; neither is this visible in any register.

Address: FBF7_4000h base + 1ACh offset + (60d × i), where i=0d to 6d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CMD					0		ACMD				0		CH_TRIG	GL_TRIG_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DEFAULT_SYNC	DDSR_SHIFT_SEL	DDSR_CLR	DDSR_CLK_SEL	CMDTR_SW_CTRL	0	SYNCHRO_OVF_IE	0	CMDTR_NWRT_IE	CMDPR_BZY_IE	DDSR_RDY_IE	0		CMD_TYPE		
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

PSI5S_E2SCR_CH*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–5 CMD	Bits for the ECU-to-sensor "CMD" format. They refer to the command send to the UART Tx logic, when a "1" is shifted from the DDSR register **NOTE:** This bit is writable only in PS_CONFIG mode.

Table continues on the next page...

PSI5S_E2SCR_CH*n* field descriptions (continued)

Field	Description
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–11 ACMD	Bits for the ECU-to-sensor "ACMD" format. They refer to the command send to the UART Tx logic, when a "0" is shifted from the DDSR register. **NOTE:** This bit is writable only in PS_CONFIG mode.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CH_TRIG	This controls that the DDSR of the specific channel is triggered locally. This bit controls the shifting when PS_E2SCR_CHn[GL_TRIG_SEL] is 0. **NOTE:** This bit is also writable in PS_NORMAL mode. 0 Stop the DDSR from shifting the commands. The counters corresponding to DDTRIG_PERR and the DDTRIG_OFFR are reset. 1 Start the "shift trigger" of the DDSR.
15 GL_TRIG_SEL	This bit controls whether the DDSR shift trigger is a global trigger or a local channel trigger. **NOTE:** This bit is writable only in PS_CONFIG mode. 0 The DDSR shift trigger is a local channel trigger dependent on the bit CH_TRIG. 1 The DDSR shift trigger is a global triggerdependent on PS_GLCR[GL_DDSR_TRIG].
16 DEFAULT_SYNC	Decides the value to which the DDSR should default to when PS_E2SSR_CHn[DDSR_RDY] == 1. When DDSR_RDY == 1 then the default shift (whether it is CMD or ACMD) is based on the value of this bit. **NOTE:** This bit is writable only in PS_CONFIG mode
17 DDSR_SHIFT_SEL	DDSR Shift Selection: This is used to select the shift trigger for the DDSR registers. Please see Figure 57-21 for more details. 0 The shift trigger is from internal DDSR counters which are clocked from either the “gtm_trig” or the ipg_clk_ps_ddtrig. 1 The input signal “gtm_trig” directly acts as the shift trigger of the DDSR registers.
18 DDSR_CLR	Used for rejecting the contents of the DDSR register. As soon as this bit is written to "1" the contents of DDSR are immediately rejected and defaulted to "0" after the completion of the current ACMD or the CMD transmission. PS_E2SSR_CHn[DDSR_RDY] goes to 1 and value of PS_E2SCR_CHn[DEFAULT_SYNC] starts getting shifted. This bit is one shot (W1S). **NOTE:** This bit is also writable in PS_NORMAL mode.
19 DDSR_CLK_SEL	Selects which clock clocks shift logic of the DDSR, for shifting the bits for command transmission: 0 ipg_clk_ps_ddtrig Clock 1 gtm_trig is selected as the clock.
20 CMDTR_SW_CTRL	This bit decides that once the command is ready for transmission (indicated by the setting PS_E2SSR_CHn[CMDPR_BZY] to 1), then when should the actual transmission start: **NOTE:** This bit is also writable in PS_NORMAL mode.

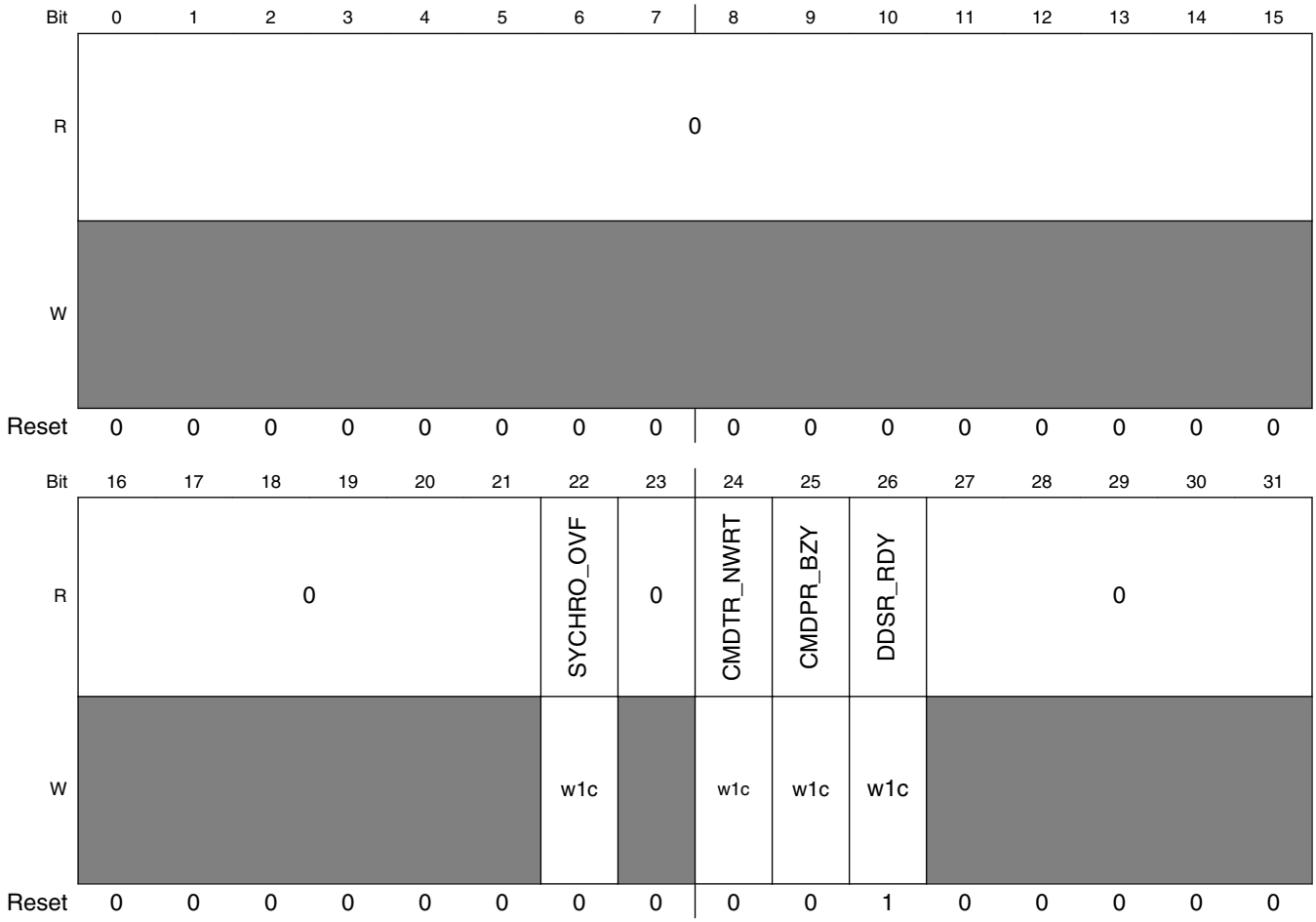
Table continues on the next page...

PSI5S_E2SCR_CH*n* field descriptions (continued)

Field	Description
	0 The actual transmission of command remains suspended, even though the command is ready in the DDSR. 1 The command transmission starts once the PS_E2SSR_CHn[CMDPR_BZY] becomes 1, and the internal shift logic is available.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 SYNCHRO_OVF_IE	Synchro overflow interrupt enable. 0 No interrupt is generated when PS_E2SSR_CHn[SYNCHRO_OVF] is set to “1”. 1 Generate an interrupt when PS_E2SSR_CHn[SYNCHRO_OVF] is set to “1”.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CMDTR_NWRT_IE	Interrupt Enable for DDSR being attempted to be written when it is still not empty i.e PS_E2SSR_CHn[DDSR_RDY] == 0 yet an attempt to write to DDSR is made. **NOTE:** This bit is also writable in PS_NORMAL mode. 0 Disable the interrupt 1 Enable the interrupt
25 CMDPR_BZY_IE	Interrupt Enable for DDSR getting updated with the internally processed complete command (with CRC, stuff bits, start bits getting appended) **NOTE:** This bit is also writable in PS_NORMAL mode. 0 Disable the interrupt 1 Enable the interrupt
26 DDSR_RDY_IE	Interrupt Enable for DDSR getting empty and ready for receiving new command. **NOTE:** This bit is also writable in PS_NORMAL mode 0 Disable the interrupt 1 Enable the interrupt
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 CMD_TYPE	See Table 57-5. The start condition (31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command; neither is this visible in any register.

57.4.40 PSI5-S ECU to Sensor Status Register (PSI5S_E2SSR_CH*n*)

Address: FBF7_4000h base + 1B0h offset + (60d × i), where i=0d to 6d



PSI5S_E2SSR_CH*n* field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 SYCHRO_OVF	This bit gets set whenever due to any reason(like too less DDTRIG period) , sync pulse of a channel comes before the previous sync pulse of the same channel has actually shifted its data. It can be used to detect sync pulse overflow. If PS_E2SCR_CHn[SYNCHRO_OVF_IE] bit is set to “1” then this can also be used to generate an interrupt on the ipi_ps_e2s_ch[n] line where “n” is the specific channel number. 0 No synch pulse overflow has occured in the specific channel. 1 Synch Pulse overflow has occured for the specific channel.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CMDTR_NWRT	Status of Command written to the DDSR:

Table continues on the next page...

PSI5S_E2SSR_CH*n* field descriptions (continued)

Field	Description
	0 Command written to the DDSR has been accepted for processing. 1 Command written to the DDSR is rejected as an attempt to write to it has been made when DDSR_RDY == 0.
25 CMDPR_BZY	Command Processing Busy Status. This bit along with the DDSR_RDY bit can be used to check the status of the command that is being processed in the DDSR. Note that when DDSR_RDY goes to "1" then this bit goes to "0". So it represents the above status conditions ONLY when DDSR_RDY is "0". 0 DDSR is busy with command processing. For this status DDSR_RDY should alse be read as "0". 1 DDSR is ready with the processed command complete with CRC/start bits/stuff bits.The transmission of the same would start if PS_E2SCR_CHn[CMDTR_SW_CTRL] == 1. For this status DDSR_RDY should also be read as "0".
26 DDSR_RDY	Status for DDSR getting empty and ready for receiving new command. 0 DDSR is not ready and is busy with either the command processing or the command transmission 1 DDSR is empty and ready for receiving a new command.
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.41 PSI5-S channel1 ECU to Sensor Downstream Data Shift Register High (PSI5S_DDSR_H_CH*n*)

Address: FBF7_4000h base + 1B4h offset + (60d × i), where i=0d to 6d

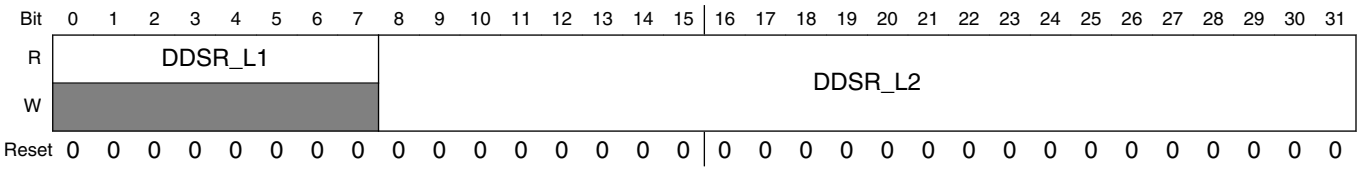
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																					DDSR_H										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5S_DDSR_H_CH*n* field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 DDSR_H	Upper High bits of the 43 bit complete DDSR register. These bits are updated by the hardware and software cannot write to these bits. They can be read by the software to check for the processed command after the command is written to the DDSR by the application , the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to "1".

57.4.42 PSI5-S channel1 ECU to Sensor Downstream Data Shift Register Low (PSI5S_DDSR_L_CH*n*)

Address: FBF7_4000h base + 1B8h offset + (60d × i), where i=0d to 6d



PSI5S_DDSR_L_CH*n* field descriptions

Field	Description
0–7 DDSR_L1	8 bits of the DDSR_L that can only be read by the Application. These 8 bits are automatically updated by the hardware . They can be read by the software to check for the processed command after the command is written to the DDSR by the application , the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to "1".
8–31 DDSR_L2	Lower 24 bits of the DDSR register that are written by the Application as well as updated by the hardware once the command processing is over. Though all the 24 bits can be wriiten by the software , yet only those number of bits as governed by PS_E2SCR_CHn[CMD_TYPE] are actually taken for processing. This register can be read by the software to check for the processed command, after the command is written to the DDSR by the application , the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to "1". **NOTE:** These bits are also writable in the PS_NORMAL mode.

57.4.43 Register Access in Different Modes

Table 57-6. Registers Access in Different Modes

Register	Access		
	Disable Mode	Config Mode	Normal Mode
PSI5-S LIN Control register 1 (PS_LINCR1)	R	R/W	R
PSI5-S LIN Interrupt Enable register (PS_LINIER)	R	R/W	R
PS_LINSR	Read value in PSI5-S modes is “don't care”. Also see below.[1].		
PSI5-S UART Mode Control register (PS_UARTCR)	R	R/W	R
PSI5-S UART Mode Status register (PS_UARTSR)	R	R/W1C	R
PSI5-S LIN Fractional Baud Rate register (PS_LINFBRR)	R	R/W	R
PSI5-S LIN Integer Baud Rate register (PS_LINIBRR)	R	R/W	R

Table continues on the next page...

Table 57-6. Registers Access in Different Modes (continued)

Register	Access		
	Disable Mode	Config Mode	Normal Mode
PSI5-S LIN Control register 2 (PS_LINCR2)	R	R/W	R
PSI5-S Buffer Data Register Least Significant (PS_BDRL)	R	R/W	R
PSI5-S Buffer Data Register Most Significant (PS_BDRM)	R	R/W	R
UPSI5-S ART Current Timeout register (PS_UARTCTO)	R	R	R
DMATXE	For PSI5-S modes, always program with 0x0000_00000 in Config Mode. Also see below.		
DMARXE	For PSI5-S modes, always program with 0x0000_00000 in Config Mode. Also see 4 below.		
Global Control Register (PS_GLCR)	R/W[2]	R/W	R/W
Global Status Register (PS_GLSR)	R/W1C	R/W1C	R/W1C
Channel_Base_Address (PS_CH_BASE_ADDR)	R	R/W	R
MRU OUTPUT BUFFER2 REGISTER0 (PS_MRU_BUF2_REG0)	R	R	R
MRU OUTPUT BUFFER2 REGISTER1 (PS_MRU_BUF2_REG1)	R	R	R
MRU OUTPUT BUFFER2 REGISTER2 (PS_MRU_BUF2_REG2)	R	R	R
MRU OUTPUT BUFFER2 REGISTER3 (PS_MRU_BUF2_REG3)	R	R	R
PSI5-S Mbox Status Irq (PS_MBOX_SR_IRQ)	R	R	R
PSI5-S Error Status Irq (PS_ERR_SR_IRQ)	R	R/W1C	R/W1C
PSI5-S Mbox select Irqn (n = 0 - 7) (PS_MBOX_SEL_IRQn)	R	R/W	R/W
PSI5-S Error Select Irqn (n = 0 - 7) (PS_ERR_SEL_IRQn)	R	R/W	R/W
PSI5-S Watchdog Frame Error Status and Watchdog Frame Errror Timestamp capture register (PS_WDGTSSR)	R	R/HC[3]	R
PSI5-S ECU to Sensor Direct Command write register (PS_DIRCMD)	R	R/W	R/W

Table continues on the next page...

Table 57-6. Registers Access in Different Modes (continued)

Register	Access		
	Disable Mode	Config Mode	Normal Mode
PSI5-S channeln message configuration register A (PS_MSGA_CHn) (n = 0-7)	R	R/W	R
PSI5-S channeln message configuration register B (PS_MSGB_CHn) (n = 0-7)	R	R/W	R
PSI5-S Mailbox status register channeln (PS_MBOX_SR_CHn) (n = 0-7)	R	R/W1C	R/W1C
PSI5-S channeln watchdog configuration register(PS_WD_CFGR_CHn) (n = 1-7)	R	R/W	R/W
PSI5-S DDSR Trigger offset register channeln (PS_DDTRIG_OFFR_CHn) (n = 1-7)	R	R/W	R
PSI5-S DDSR Trigger period register channeln (PS_DDTRIG_PERR_CHn) (n = 1-7)	R	R/W	R
PSI5-S channeln ECU to Sensor Control register (PS_E2SCR_CHn) (n = 1-7)	R	R/W	R/W
PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CHn) (n = 1-7)	R	R/W1C	R/W1C
PSI5-S channeln ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH1) (n = 1-7)	R	R	R
PSI5-S channeln ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CHn) (n = 1-7)	R	R/W	R/W

1. Register has significance in UART STANDLONE mode. This document would only mention the bit field names in the corresponding register description. For detailed bit description and the functioning of this register in UART modes(sleep,init and normal) belonging to UART STANDLONE mode, please refer to the separate documentation describing the standalone LINFLEX module.
2. Not all register bits are of this type. Please see the register bit description for more details.
3. Register Bits are automatically hardware cleared on read of any bit of this register.

57.5 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The UART compatible transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. Figure 57-4 shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

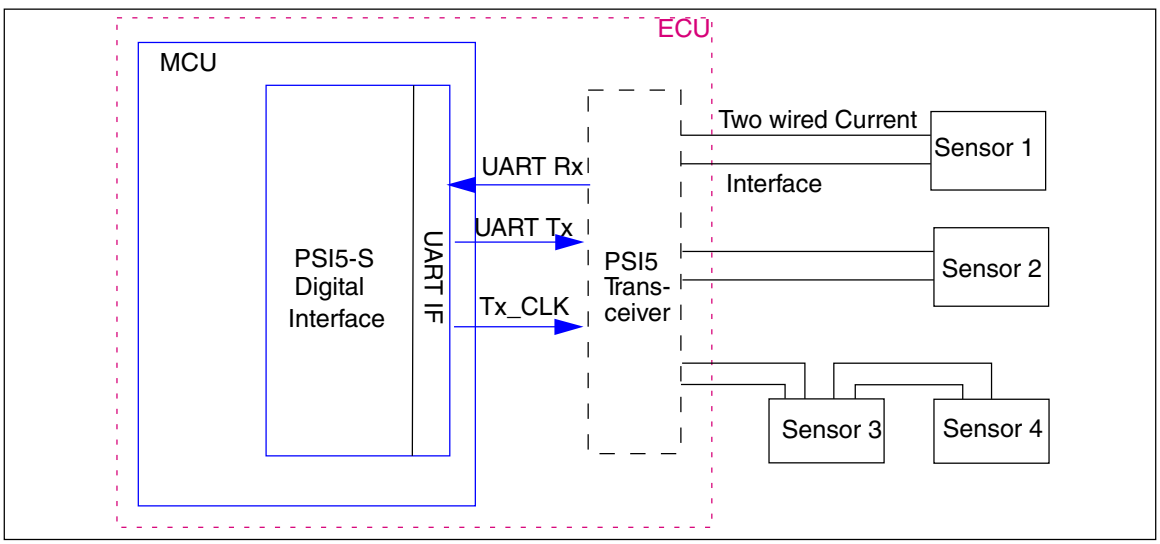


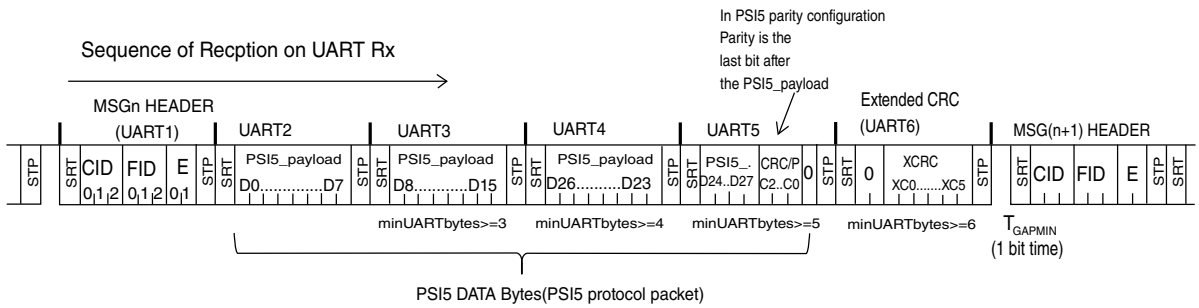
Figure 57-4. System setup (peripheral sensor connected to device)

The ECU (Electronic Control Unit) comprises PSI5-S digital interface and the UART compatible PSI5 transceiver (present externally). The PSI5-S digital interface communicates with the PSI5 Transceiver though the UART interface as shown in Figure 57-4. When conducting the Sensor To ECU communication, the external transceiver converts the sensor data to the UART packet format before sending it to the PSI5-S module. The IP extracts the PSI5 Frames from the input UART Packet.

57.5.1 PSI5-S UART Message Structure

The transceiver sends a UART message packet consisting of between three to six UART bytes which contains information to allow rebuilding of the PSI5 message as well as the message itself and has a CRC which protects the complete packet (XCRC).

Each byte is made up of one start bit, eight data bits and one stop bit - no parity is included in the UART byte. Figure 57-5 shows the PSI5-S UART packet structure.



Total Message Payload(TMPL) = header_bits(8)+PSI5_payload(8-28bits) +CRC/P(3/1 bit)+XCRC(6bits)
minUARTbytes=CEIL(TMPL/8) where CEIL gives the largest integer greater than or equal to TMPL/8
PSI5_payload is the value obtained from PS_MSGA_CHn[Fn_payload] register bits.
XCRC(Extended CRC) can occupy any byte from UART3 to UART6 depending on the number of bytes received.

Figure 57-5. PSI5-S Packet Structure

The UART packet is made up of the following:

- MSGnHEADER: This is the message header byte which includes the indication of which channel the PSI5 message was received on (3-bit CID), the slot on that channel in which the message was received (3-bit FID) and transceiver error indication bits (2-bit E)
- PSI5 DATA bytes - minimum one, maximum four, depending on the data payload region of the PSI5 message (8 - 28 bits) and whether parity (1-bit) or CRC (3-bit) is configured for the PSI5 message. In the above example shown in Figure 57-5 it is taken as 28 bits and 4 UART bytes are shown(UART2-UART5). Messages having CID and FID both as "0s" do not have the CRC/Parity field in them. Such messages only have the message header, payload and the xcrc fields.
- XCRC- 6-bits - This is the extended CRC which protects the UART packet. Note that any bits between the end of the PSI5 message payload and the XCRC are 'stuffed' with '0's. This can also be seen in Figure 57-5. For details about XCRC handling please refer PSI5-S XCRC handling.
- Example to calculate the minUARTbytes (for messages NOT having CID and FID both as 0):
  - Fn_payload = 14 with PSI5 CRC option: TMPL = 8+14+3+6=31, minUARTbytes=CEIL(3.8)=4.

    These input data stream is interpreted as **UART1**(header),**UART2**(D[0:7]),**UART3**(D[8:13],CRC[2:1]), **UART4**(CRC[0],0,XCRC[0:5]).
  - Fn-payload = 13 with PSI5 Parity option: TMPL=8+13+1+6=28,minUARTbytes=CEIL(3.5)=4.

These input data stream is interpreted as
UART1(header),**UART2**(D[0:7]),**UART3**(D[8:12],Paritybit,00),
UART4(00,XCRC[0:5]).

Note the "0" stuffing between the parity bit and the XCRC. The XCRC is always in the last 6 bits of the last byte.

- Example to calculate the minUARTbytes (for messages having CID and FID both as 0):
  - F0_payload = 8: TMPL = 8+8+0+6=22, minUARTbytes=CEIL(2.75)=3

    These input data stream is interpreted as

    UART1(header),**UART2**(D[0:7]), **UART3**(00,XCRC[0:5]).
  - F0_payload = 10: TMPL = 8+10+0+6=24, minUARTbytes=CEIL(3)=3

    These input data stream is interpreted as

    UART1(header),**UART2**(D[0:7]), **UART3**(D[8:9],XCRC[0:5]).

57.5.2 PSI5-S Received Message handling

The message handling in the IP can be divided in three parts:

- PSI5 message extraction:

  This decides when a UART packet has been received with the correct header byte, and then how to extract the information from this UART packet form a complete PSI5 message.
- PSI5 "recoverable" and "unrecoverable messages":
- This differentiates the PSI5 messages received above, into "recoverable" and "unrecoverable" messages.
- PSI5 message storage and Mailbox transfer:

  This ensures the transfer of the PSI5 message to the correct location in the system RAM, once the PSI5 message has been correctly differentiated into "recoverable" and "unrecoverable" messages.Following sections detail the above 3 parts:

57.5.2.1 PSI5-S Message extraction

Once a UART start bit is detected after the idle timeout (the idle time between message packets on the UART interface i.e. the number of bit times between message packets in the range of at least 1 to 16, configurable in the UART module through the PS_UARTPTO register) has occurred, the UART bytes are subsequently sampled. They are being received by an internal state machine in the MRU, which checks for the formation of a complete UART packet, its coherence and extraction of the PSI5 message from the UART packet. Also please see Figure 57-6 for further understanding of this section.Each UART packet that is received can be considered as "recoverable" or "unrecoverable". Demarcation between the two is explained subsequently. Following are the sequence of steps performed as each UART byte is being received:

1. There is a 6 byte UART message input buffer into which the UART bytes from the UART Rx register are written as they are received. A 3-bit UART message counter is incremented on the reception of each byte in UART message packet i.e. each UART byte received between idle time counter indications.
2. All the subsequent UART bytes in the UART packet are stored internally till any of the following events occurs:
   - UART Timeout i.e. bus idle time is detected.
   - Framing Error from the UART.
   - Number of UART bytes that are received as reported through the UART message counter described in "1" above, exceed 6.
   - STOP mode request mode is asserted. Please see Stop mode for further details.
3. As soon as any of the events described in pt. 2 above occur, it is assumed that a complete UART packet has been received. The XCRC is then calculated immediately on the received UART packet. Please see PSI5-S XCRC handling for details about XCRC. If there is an XCRC error the message is assumed to be "unrecoverable" and it is sent to $MBOX_0$,LOC_1.The timestamp of the message is captured as is described in TimeStamp. The watchdog error is captured as is described in PSI5-S Watchdog Operation.
4. If there is NO XCRC error, then the CID and FID are extracted from the header byte, and the configured length of the message is determined from the configuration in PSI5 Channel*n* Message Configuration Register A. Following steps are performed.
   - The FID and the CID from the header byte is transferred directly to the PSI5 Frame status word.

- The transceiver error flags (E0,E1) are transferred directly to the PSI5 Frame status word
- The message is then analyzed to see whether it is "LEGAL" or "ILLEGAL".
- Following are the conditions for a "LEGAL" message:

  When the CID and the FID of the received message are both "0s" and PS_MSGA_CH0[F0_byte] is NOT "0" OR

  When the data of an Asynchronous Enabled channel is received with FID = 0 and PS_MSGA_CH_{CID}[F0_byte] is NOT "0" OR

  When the data of a Synchronous Enabled channel is received with legal FIDs(0 to 5) and if the received FID is such that the corresponding PS_MSGA_CH_{CID}[F_{FID}_byte] is NOT 0
- Any received message that does not satisfy the above condition is considered as "ILLEGAL". Illegal Messages are also treated as "unrecoverable" messages and are sent to $MBOX_0$,LOC_1.

  Example can be a message with CID as 0 but FID as 1(illegal because Channel0 is asynchronous and can only have FID equal to 0 messages) or a message received for an enabled synchronous channel but having an FID equal to 7(illegal because FID as 7 is not a legal Frame ID) etc.
- The timestamp of a Legal/Illegal message is captured as is described in TimeStamp. The watchdog error is captured as is described in PSI5-S Watchdog Operation.
- The number of UART bytes received in the UART message packet is compared with the number specified in Message Configuration Register A for the channel indicated by CID and FID in the header byte. If the number of bytes indicated by the message counter does not match PS_MSGA_CH_{CID}[F_{FID}_byte] field, then the status bit N_Err within the PSI5 Frame status word is set. The above "number of bytes received" check is not carried out for an "illegal message".

5. The PSI5 message payload is checked using the PSI5 CRC or parity as per the configuration. Whether the CRC or parity check is correct or not, the PSI5 message payload is written to the PSI5 Sensor Data word and the CRC or parity is written to the PSI5 Frame Status word. Note that the messages which have CID and FID both as "0", are not checked for CRC or parity error. The CRC/Parity error status bits remain "0" for such messages.This above check is NOT done for "illegal messages".

57.5.2.2 PSI5-S "Recoverable and Unrecoverable" Messages

Based on PSI5-S Message extraction, a message is considered "unrecoverable", if the XCRC is incorrect or it is an "illegal message", else the message is considered "recoverable". "Unrecoverable" message indicates that an error has been detected during the assembling phase of the UART packet i.e., during the formation of the UART packet from the individual UART bytes. "Unrecoverable" messages are always stored in PSI5 mailbox $MBOX_0$,LOC_1 in system RAM. "Recoverable" messages are always stored in the Mailbox location pointed out by the CIDn and the corresponding FID fields.

Figure 57-6 describes the treatment of the "Recoverable" and "Unrecoverable" messages. It also details the setting of the various error flags. The various cases for a new PSI5 message reception are:

- Idle Timeout occurence.
- Framing Error Occurence.

  As soon as a framing error has occured , all the UART packets sampled till that instance are assembled to form a PSI5 packet and are transferred to the MRU buffer. The state machine starts sampling the next UART byte only after the occurrence of an idle Time Out from the UART or after a reset or if the mode of the IP is changed (PS_NORMAL to PS_CONFIG/PS_DISABLE/UART_STANDALONE and then back to PS_NORMAL) or if a stop mode is entered and then exited.

- Number of UART bytes recieved exceed 6.

  In this case, any UART bytes recieved after the 6th byte, are totally rejected by the internal PSI5-S state machine. The state machine starts sampling the next UART byte only after the occurrence of an idle Time Out from the UART or after a reset or if the mode of the IP is changed (PS_NORMAL to PS_CONFIG/PS_DISABLE/UART_STANDALONE and then back to PS_NORMAL) or if a stop mode is entered and then exited.

- STOP mode entry:

  Please refer Stop mode). Stop mode entry causes the flushing of the internal PSI5 message buffer and this event is treated like new message completion. Exiting from the stop mode resets the state machine and it becomes ready to respond to the next UART packet. The next UART packet is taken as the header byte of a new PSI5 message.The stop mode, as an event for a new PSI5 message formation is not shown in Figure 57-6.

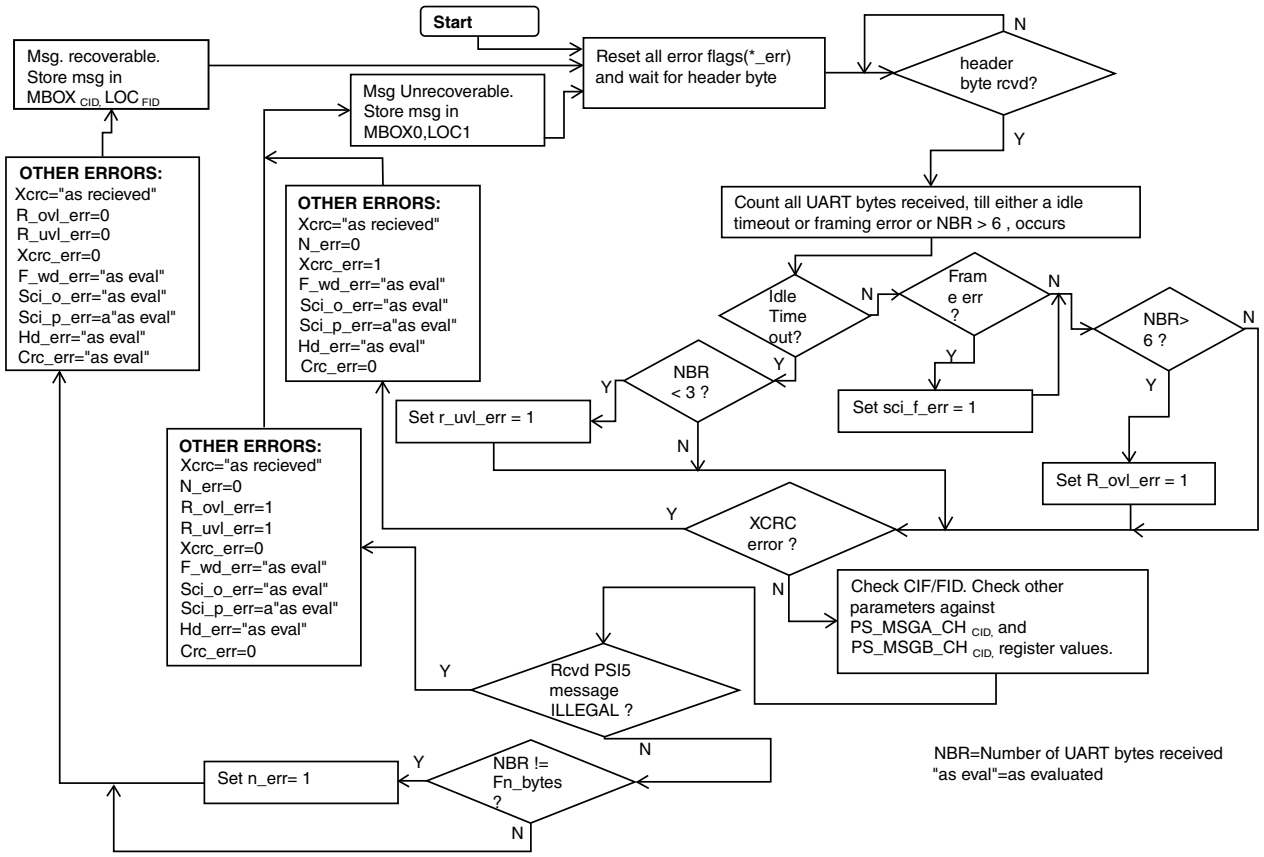


Figure 57-6. PSI5-S message handling and Error Flags

Once the message is correctly deciphered as "Recoverable" or "Unrecoverable" and the various error flags are set, then the Message is finally transferred to the MRU buf2. Once the corresponding message is read from the MRU buf2, the bits in the "PS_MBOX_SR_CH[n]" register are set appropriately. This aids the application to correctly read the concerned message from the Mailbox. PSI5-S Message Storage and Mailbox Transfer describes the PSI5 Message Storage and detection in detail.

57.5.2.3 PSI5-S Message Storage and Mailbox Transfer

Once the message has been correctly deciphered as "recoverable" or "unrecoverable" and the various status bits are set, it is then transferred to the MRU output buffer. The MRU output buffer is a 2 stage buffer differentiated as MRU Buf1 and MRU Buf2. The MRU buf1 is composed of 3,32 bit registers while the MRU buf2 is composed of 4, 32bit registers. Figure 57-7 shows the structure of MRU buf1. It is composed of three internal registers. These registers are not in the addressable space of the IP. When the internal state machine has correctly decoded the PSI5 message, it is transferred to MRU buffer1. It serves to hold the PSI5 message until it is transferred to MRU buf2.

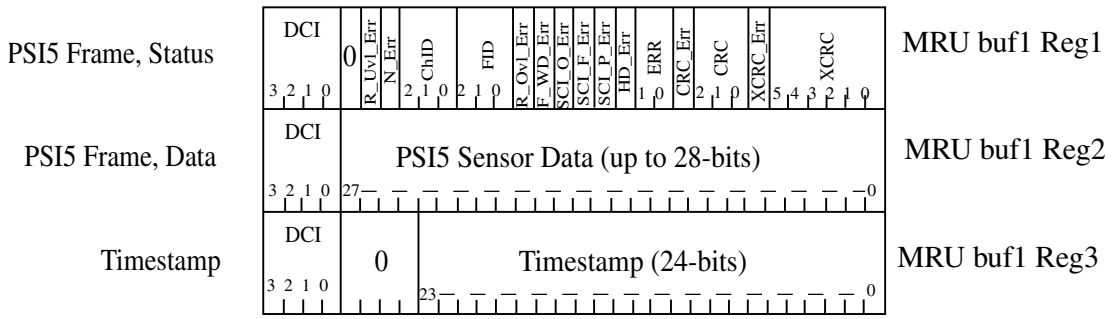


Figure 57-7. MRU buf1 structure

Figure 57-8 shows the structure of MRU buf2. MRU buf2 has one extra location as compared to MRU buf1. This extra location (MRU buf2 Reg0) contains the address of the Mailbox location, where the message contained in the rest of the three locations of MRU buffer2 is to be placed. This specific MRU buf2 structure aids the DMA to perform linked scatter-gather data transfer from the IP. The extra location in the MRU buf2 is automatically updated by the hardware and provides the destination address for the message. This destination is the Mailbox Address at which the message (stored in MRU buf2 (Reg1-Reg3) is to be transferred by the DMA. This destination address is calculated by the IP based on ChID in the message header (or ChID0 in the case of an unrecoverable message), the FID in the message header and Base address of the mailbox in system RAM (provided by CH_BASE_ADDR) register.

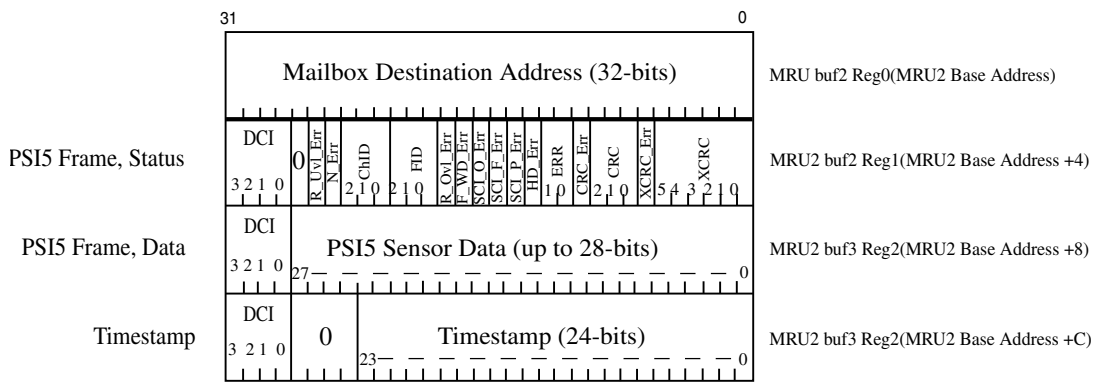


Figure 57-8. MRU buf2 structure

MRU buf2 is in the addressable space of the IP and is read by the DMA when the corresponding DMA request is asserted. The PSI5 message is transferred from the MRU buf1 to the MRU buf2 only when the MRU buf2 has been read completely by the DMA, i.e., all the 4 locations of the MRU buffer2 have been read by the DMA, for the corresponding PSI5 message.

Following are the sequence of operations that are performed, before the message in MRU buf2, gets ready to be accessed by the DMA:

1. The IP decodes the message as detailed in PSI5-S Message extraction and PSI5-S "Recoverable and Unrecoverable" Messages.
2. The corresponding message is available in MRU buf1 along with the various status bits latched internally in the IP. Externally, no interrupt is generated and neither are any status bit updated, when the message is transferred to MRU buf1.When the message is transferred to MRU buf1 a 3 bit DCI (data consistency indicator is added). The DCI is a 3-bit counter that is counted on a reception of each new message in MRU buf1.
3. The IP checks whether MRU buf2 is empty. If it is empty, then the MRU buf1 "Reg1 to Reg3" contents are transferred to the MRU buf2 "Reg1 to Reg3". The corresponding interrupts (except the Message Read from MRU) are also asserted based on the Interrupt Enable settings. Based on the PSI_CH_BASE_ADDR, the CID, FID values and whether an "unrecoverable message" has been received, the IP calculates the contents of MRU buf2 Reg0 register.
    - If CID == 0 then the "MRU buf2 Reg0" contains the Mailbox Location corresponding to MBOX0,LOC0.
    - If "unrecoverable message" is received then "MRU buf2 Reg0" contains the Mailbox Location corresponding to MBOX0,LOC1.
    - In all other cases "MRU buf2 Reg0" contains the Mailbox location corresponding to the $MBOX_{CID}$,LOC_{FID}.
4. Once the calculations in (3), are done the IP asserts the DMA request for Access1. In DMA Access 1, the DMA reads the first location (MRU buf2 Reg0) of the MRU buf2. This single 32-bit access writes the content of MRU buf2 Reg0 into the destination address for the DMA channel performing the second access (Access 2). Access 2 is initiated by the DMA once the Access1 transfer is complete. Access 2 is composed of 3,32-bit accesses of MRU buf2 and it writes the reconstructed PSI5 message along with its status and timestamp to the mailbox, starting at the destination address provided by Access 1.In Access2 the DMA reads the MRU buf2 Reg1,Reg2 and Reg3 in that order.
5. The application can read the PS_MBOX_SR_CH[n] register to get an indication to which mailbox location (or locations) the DMA would transfer the message. Note that "DMA transfer complete" bit/interrupt of the DMA indicates to the application, that the message read by the DMA from the IP has actually been written to the Mailbox.

6. The application should clear the corresponding "Fn_read" bits in the PS_MBOX_SR_CH[n] register once the application has read the corresponding message from the Mailbox. If a PSI5 message gets written to a Mailbox location, with "Fn_read" bit still set at the time at which the DMA has started the Acces1, then the corresponding "Fn_OV" bits are set in the PS_MBOX_SR_CH[n] register.
7. Reference to (3) above, if the MRU buf2 has a pending message and the MRU buf1 also has a message that is ready to be transferred to the MRU buf2 and then a new message gets ready to be written to MRU buf1, then the previous message in MRU buf1 gets overwritten by the new message. The MRU overwrite interrupt is also generated with the setting of the corresponding register bit in the PS_GLSR register. PS_GLSR register contains the CID and the FID of the message that got overwritten by the new message in MRU buf1.
8. When writing an unrecoverable message to the PSI5 mailbox CID0, with the XCRC_ERR bit set, the timestamp that will be used will be the one which is captured at the time XCRC_ERR bit is set. When the unrecoverable message is due to an "illegal message", then the timestamp used would be the one which is captured on the reception of any message i.e., always captured at the end of the header byte of every message.
9. In Asynchronous mode, FID will always be configured to be "0" and the module will use the PSI5 Mailbox for the channel (identified by CID in header) as a ring buffer, incrementing the destination address for each message received and saving up to the last six messages received. The bits of the PS_MBOX_SR_CH[n] register will still indicate the status of each location of the Ring Buffer. The software has to still clear the six bits of this register when it reads the Mailbox location else the overwrite bits would be set in the PS_MBOX_SR_CH[n] register. The message identification mechanism allows software to identify the latest message received

57.5.2.4 PSI5-S Data Payload CRC Recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

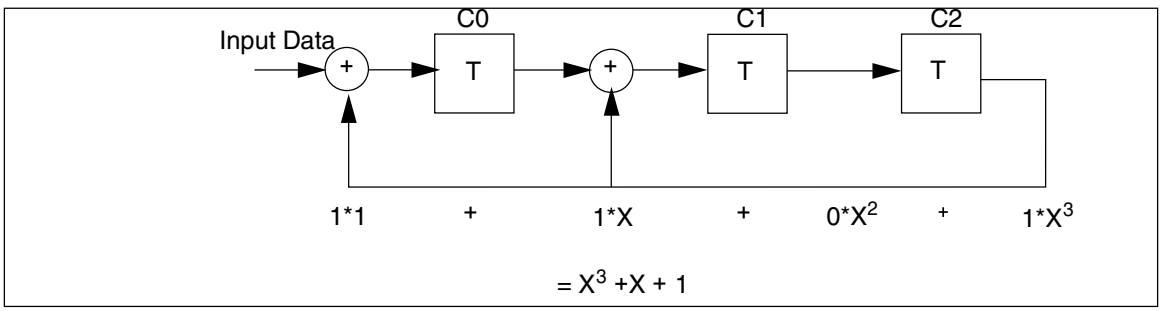


Figure 57-9. CRC calculation scheme

Table 57-7. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 57-8. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 57-9. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 ® Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

57.5.2.5 PSI5-S XCRC handling

The XCRC calculation is triggered on any of the following events (which also indicate the reception of a new UART packet):

- UART idle timeout
- number of UART bytes exceed 6
- UART Framing error
- STOP mode assertion.Please see Stop mode.

When any of the above events occur, the internal XCRC calculator becomes active. Each UART byte that is received, passes through the XCRC calculator on a bit by bit basis, starting from the header byte to the expected XCRC byte. The XCRC is assumed to be the last 6 bits of the last UART byte that is received before the occurrence of any of the above events. Figure 57-10 shows the XCRC calculation algorithm.

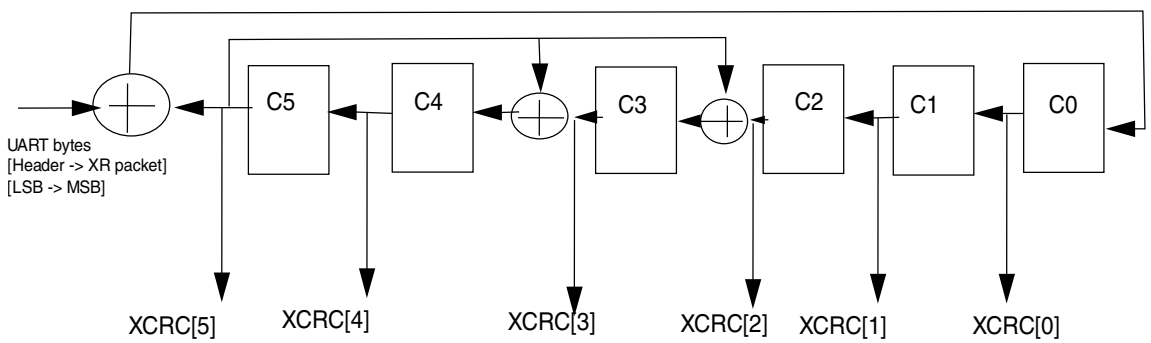


Figure 57-10. XCRC calculation

The UART bytes are shifted starting from the received header to data packets to the XCRC packet. Each of these UART bytes is shifted LSB first into this XCRC calculator. Once all the bytes as denoted by the Fn bytes have been received and the Fnth packet (XCRC) has been shifted completely through this XCRC calculator and if XCRC[5:0] is equal to "0" then it denotes that the XCRC check has passed else there is an XCRC error and the XCRC_ERR bit is set.

The CRC polynomial used is "x6+x4+x3+1" with a binary initialization value of XCRC[5:0] = "010101".

57.5.3 ECU-to-sensor communication

The IP send the commands for the ECU to Sensor communication in terms of UART bytes. It expects the external transceiver to correctly interpret the commands and then generate the appropriate logic 0 or logic 1 on the PSI5 bus. Following are the various PSI5-S ECU to Sensor commands supported by the IP, as shown in Figure 57-11.

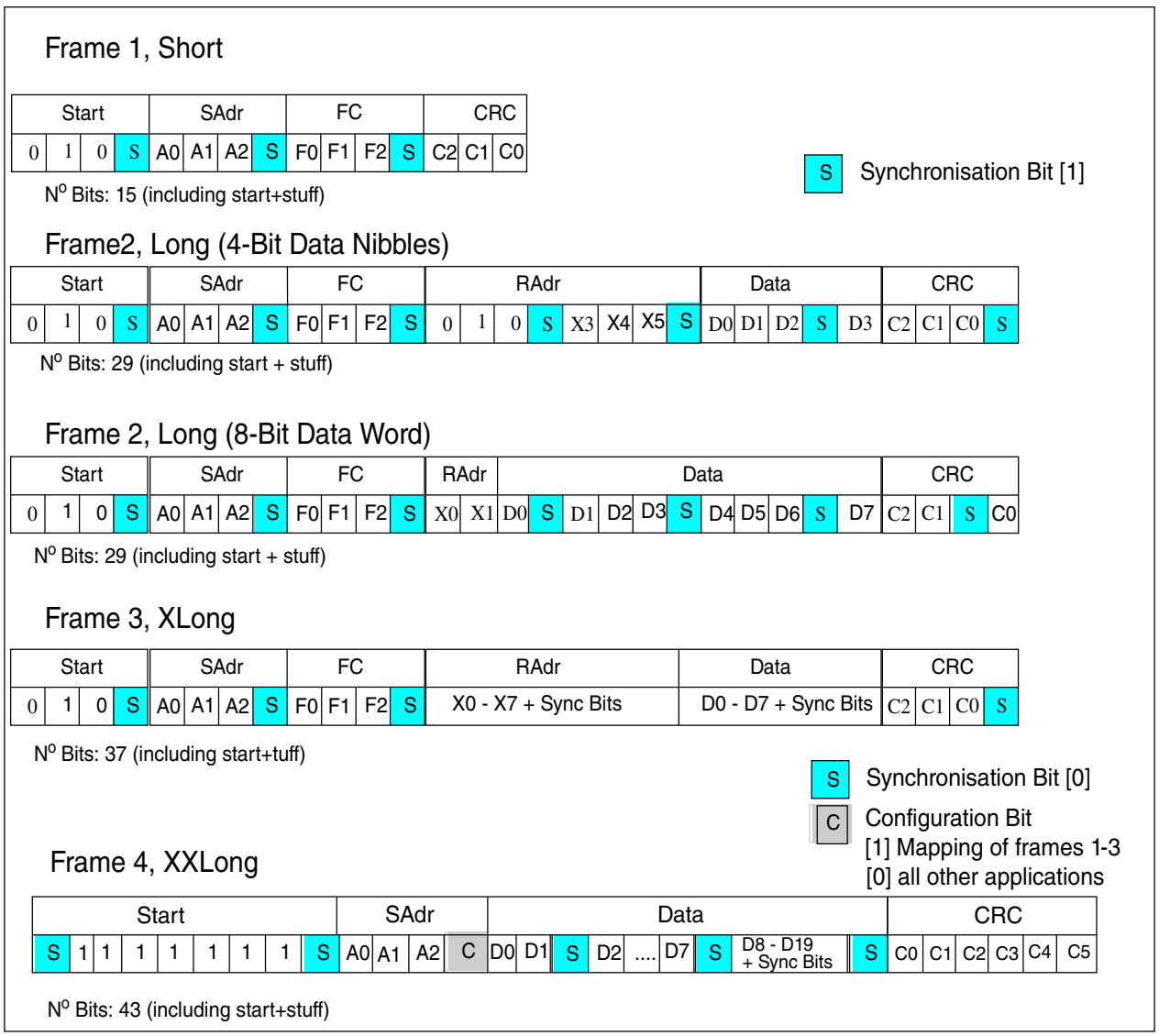


Figure 57-11. Standard Frame Formats Supported

- **Short:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- **"010",S, "A0,A1,A2",S, "F0,F1,F2",S, "C2,C1,C0"** where "010" is the start sequence "S" is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes and "C2-C0" is the CRC. Please refer Figure 57-11 for the frame structure.

- The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of "111(C2:C0)".
- The transmitter extends the data bits by three zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
- When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]….PS_DDSR[5],000) with PS_DDSR[0] being fed first in the CRC calculator.

- **Long(4bit Data Nibble)**: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- **"010",S, "A0,A1,A2",S, "F0,F1,F2",S, "X0,X1,X2",S, "X3,X4,X5, S, "D0,D1,D2", S,"D3,C2,C1",S, "C0"** where "010" is the start sequence "S" is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes,"X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Please refer Figure 57-11 for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of "111(C2:C0)".
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,X0..X5,D0…D3,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]….PS_DDSR[15],000) with PS_DDSR[0] being fed first in the CRC calculator.
- **Long(8bit Data Word)**: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - **"010",S, "A0,A1,A2",S, "F0,F1,F2",S, "X0,X1,D0",S, "D1,D2,D3, S, "D4,D4,D6", S,"D7,C2,C1",S, "C0"** where "010" is the start sequence "S" is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes,"X0-X1" is the Range Address, "D0-D7" is the data and "C2-C0" is the CRC.Please refer Figure 57-11 for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of "111(C2:C0)".

- The transmitter extends the data bits by three zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,X0..X1,D0…D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
- When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]….PS_DDSR[15],000) with PS_DDSR[0] being fed first in the CRC calculator.

- **X-Long:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - **"010",S, "A0,A1,A2",S, "F0,F1,F2",S, "X0,X1,X2",S, "X3,X4,X5", S, "X6,X7,D0",S, "D1,D2,D3",S, "D4,D5,D6",S,"D7,C2,C1",S, "C0"** where "010" is the start sequence "S" is the stuff bit(1),A0-A2 is the sensor address,F0-F2 are the functional codes,"X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Please refer Figure 57-11 for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of "111(C2:C0)".
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,F0..F2,X0…X7,D0…D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]….PS_DDSR[21],000) with PS_DDSR[0] being fed first in the CRC calculator.
- **XX-Long:** It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - **"011111110", "A0,A1,A2", "C", "D0,D1",S, "D2,D3,D4" D5,D6,D7, S, "D8,D9,D10,D11,D12,D13", S,"D14,D15,D16,D17,D18,D19",S, "C0,C1,C2,C3,C4,C5"** where "011111110" is the start sequence "S" is the stuff bit(0),A0-A2 is the sensor address, "C" is the configuration bit, "D0-D19" is the data and "C0-C5" is the CRC. Please refer Figure 57-11 for the frame structure.
  - The CRC polynomial used is $x6 + x4 + x3 + 1$ with a binary CRC initialization value 010101(C5:C0).
  - The transmitter extends the data bits by six zeros (as MSBs).

- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0…A2,D0..D19,"000000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
- When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0],PS_DDSR[1]….PS_DDSR[23],000000) with PS_DDSR[0] being fed first in the CRC calculator.

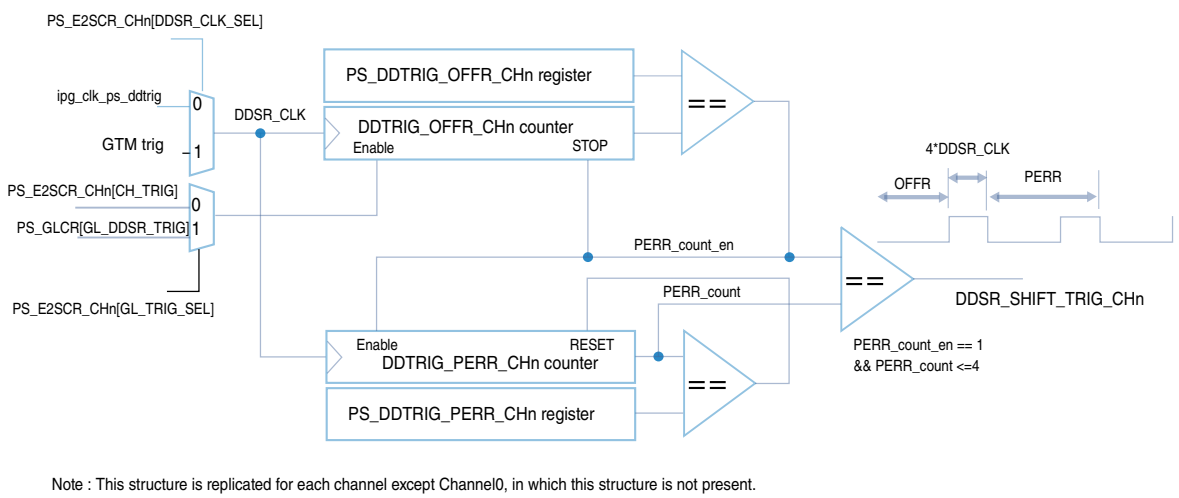
57.5.3.1 PSI5-S ECU to Sensor Command Handling

The ECU-to-Sensor commands can be written to the PSI5 IP in two ways:

1. Direct writing for the special Transceiver commands: This can be used by the software to directly write the transceiver/diagnostic commands without using the internal sync pulse generators. Following procedure should be adopted for this:
   - Application configures the PS_GLCR[DIRCMD_LEN] register for deciding the length of the direct command that is to be written. The length can be 1, 2, or 4 bytes.
   - Application will check for the status flag PS_GLSR[DIRCMD_RDY]. A “1” here would mean that the PS_DIRCMD register is empty and ready to accept the Direct write command. Trying to write to PS_DIRCMD while DIRCMD_RDY is “0”, will be rejected without any information. If the PS_GLCR[IE_DIRCMD_RDY] bit is set to “1” then an interrupt will also be generated on the “ipi_ps_glbl” interrupt line, whenever the PS_GLSR[DIRCMD_RDY] bit goes to “1”.
   - The application then writes the Direct Command to the PS_DIRCMD register. Depending on the PS_GLCR[DIRCMD_LEN] register, only the specific bytes are considered for transmission. Other bytes are ignored.
   - A “w1c” to the PS_GLSR[DIRCMD_RDY] bit would queue the above direct command for UART transmission with the highest priority; and thereafter as soon as any pending current transmission is complete, the internal state machine would start the transfer of this direct command to the UART Tx line. All other channels would internally buffer their pending DDSR transmission and resume only once the above transmission is over.
   - Once the Direct transmission is complete the PS_GLSR[DIRCMD_RDY] status bit would go to “1” and if the corresponding interrupt is enabled in the PS_GLCR register, then an interrupt will also be generated on the “ipi_ps_glbl” interrupt line.
2. Writing to the DDSR (Downstream Data Shift Register):

In this case the application writes the basic command to the DDSR_L and the IP will add the start bits/start condition/CRC/stuff bits before sending this command to the UART Tx FIFO. Following are the details:

- There is one DDSR per channel. The default value of DDSR is dependent on the PS_E2SCR[CMD_TYPE] value. DDSR is composed of two sub registers - DDSR_H (Higher) and DDSR_L (Lower). The application can only write to DDSR_L. The number of bits that can be written in the DDSR_L is dependent on the value of PS_E2SCR_CHn[CMD_TYPE] bit values. Note that Channel0 does not have any DDSR, as it does not support any ECU-to-Sensor communication.
- The "shift trigger" of the DDSR is governed by the values programmed in the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] register and the frequency of ipg_clk_ps_ddtrig clock.
- Once the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] are correctly programmed and each channel is programmed to respond to the selected shift trigger (local trigger or global trigger is decided via the bit PS_E2SCR_CHn[GL_TRIG_SEL]), the software has to write to the Global DDSR trigger bit, PS_GLCR[GL_DDSR_TRIG] or the PS_E2CR[CH_TRIG] bit, so as to synchronously start the DDSR triggers of each channel or locally start the DDSR shift triggers of each channel. The DDSR trigger logic gets the trigger signal immediately but it responds after a delay as programmed in the PS_DDTRIG_OFFR_CH[n] register. The shift trigger period is governed by the value programed in the PS_DDTRIG_PERR_CH[n] register. Please see Figure 57-12 for the structure of the logic that generates the internal DDSR shift triggers.



Note : This structure is replicated for each channel except Channel0, in which this structure is not present.

Figure 57-12. DDSR internal shift trigger generator

- The number of bits of DDSR_L that can be written by the application, is dependent on the value of PS_E2SCR_CHn[CMD_TYPE] value.
- When PS_E2SSR_CHn[DDSR_RDY] == 1, then it indicates that the DDSR is empty and is ready to accept a new command. If the PS_E2SCR_CHn[DDSR_RDY_IE] == 1, then this results in the generation of interrupt on the "ipi_ps_e2s_ch[n]" line.
- Attempting to write to DDSR while PS_E2SSR_CHn[DDSR_RDY] == 0 results in the rejection of the new command and setting of the PS_E2SSR_CHn[CMD_NWRT] bit. If the PS_E2SCR_CHn[CMDTR_NWRT_IE] == 1, then this results in the generation of interrupt on the "ipi_ps_e2s_ch[n]" line.
- The application then writes the command to the PS_DDSR_L and writes a "w1c" to the PS_E2SCR_CHn[DDSR_RDY]. This indicates to the internal logic that the command is now ready for further processing. The clock that is used for shifting the contents of the DDSR can be configured through the PS_E2SCR_CHn[DDSR_CLK_SEL] register.
- Depending on the value of PS_E2SCR_CHn[CMD_TYPE], the internal logic adds the start bits/start condition/CRC/bit stuffing and updates the DDSR with the complete command, ready for transmission.
- Once this happens the PS_E2SSR_CHn[CMDPR_BZY] becomes "1". If the PS_E2SCR_CHn[CMDPR_BZY_IE] == 1, then this results in the generation of interrupt on the "ipi_ps_e2s_ch[n]" line. If PS_E2SCR[CMDTR_SW_CTRL] == 0, then the application can check the final value of the command in DDSR, that would actually be transmitted. Once the checking is complete, it can put the PS_E2SCR_CHn[CMDTR_SW_CTRL] to "1", to start the actual transmission. By default PS_E2SCR_CHn[CMDTR_SW_CTRL] is set to "1". Note that the start condition is not a part of the DDSR and its transmission is dependent on the value of the PS_E2SCR_CHn[CMD_TYPE] and is automatically taken care of, by the IP.
- Each bit of the command in DDSR, if it is "1", results in PS_E2SCR_CHn[CMD] being transferred to the UARTTX buffer or if it is "0", results in PS_E2SCR_CHn[ACMD] being transmitted to the UART-TX buffer. The "CMD" and the "ACMD" governs as to how the transceiver has to respond when it receives these values. These will specify to the transceiver whether it should send a short or long SYNC pulse on the PSI5 bus when the application wishes to write to the sensors.

- Once the complete command has been transmitted, the PS_E2SSR_CHn[DDSR_RDY] goes to "1" while PS_E2SSR_CHn[CMDPR_BZY] goes to "0".
- Once the PS_E2SSR_CHn[DDSR_RDY] goes to "1", then default "1s" or default "0s" are transmitted, depending on the value of PS_E2SCR_CHn[CMD_TYPE].
- When PS_E2SSR_CHn[DDSR_RDY] == 1, then it is possible to abort the current transmission by doing a "W1C" of the PS_E2SCR[DDSR_CLR] bit. As soon as this is done, the DDSR is reset to "0", PS_E2SSR_CHn[DDSR_RDY] goes to "1" while PS_E2SSR_CHn[CMDPR_BZY] goes to "0". Once the PS_E2SSR_CHn[DDSR_RDY] goes to "1", then default "1s" or default "0s" are transmitted, depending on the value of PS_E2SCR_CHn[CMD_TYPE]. Note that doing this, "W1C" does not have any effect on the CMD/ACMD transmission that is already under progress in the UART.

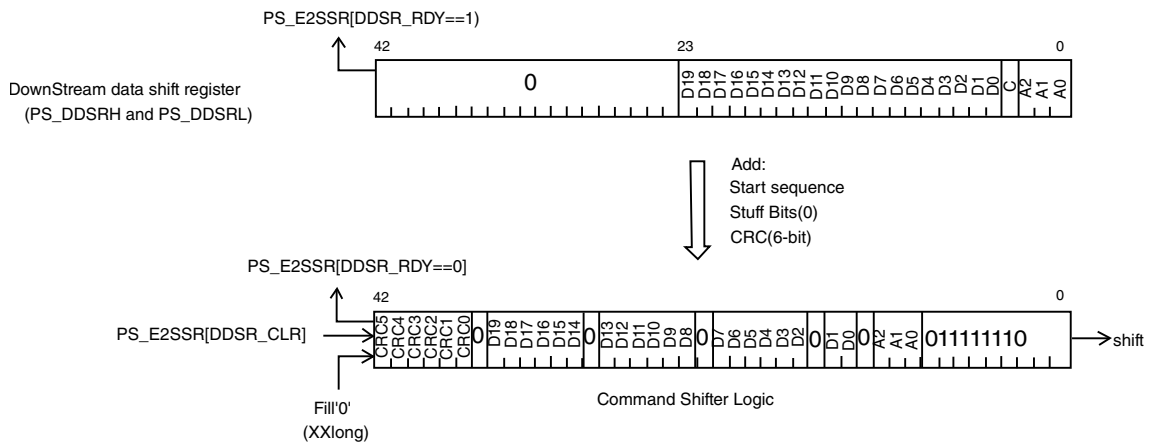


Figure 57-13. PS_DDSR Sensor Command (Ex: XXLong Frame)

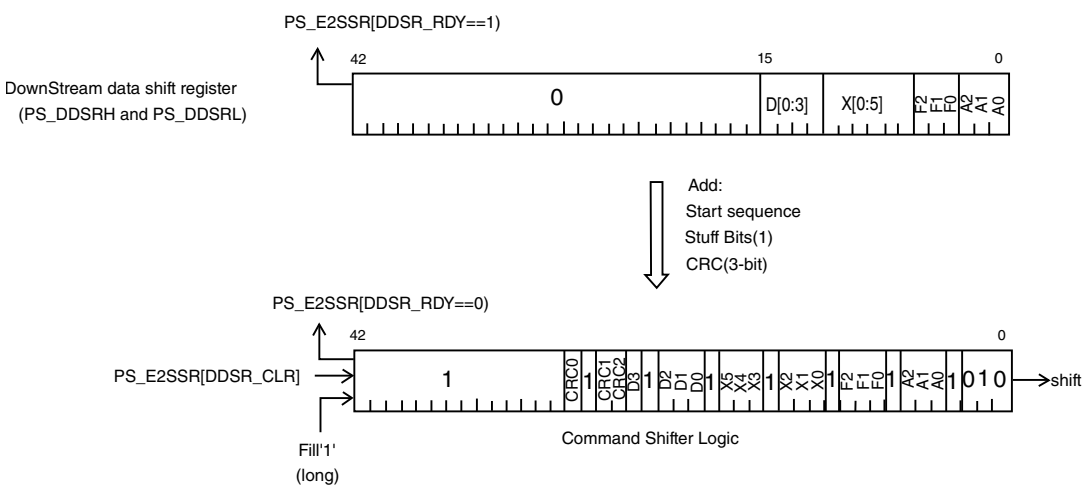
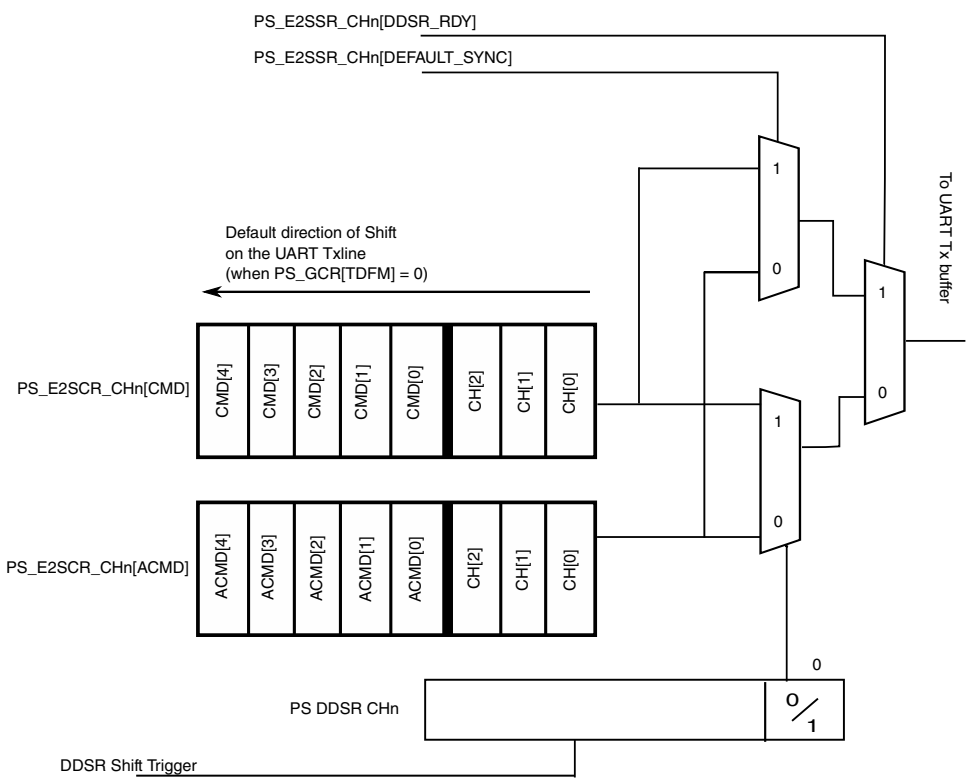


Figure 57-14. PS_DDSR Sensor Command (Ex: PSI5 LONG format)


PS_E2SSR_CHn[DDSR_RDY]
PS_E2SSR_CHn[DEFAULT_SYNC]
Default direction of Shift
on the UART Txline
(when PS_GCR[TDFM] = 0)
To UART Tx buffer
PS_E2SCR_CHn[CMD]
CMD[4]
CMD[3]
CMD[2]
CMD[1]
CMD[0]
CH[2]
CH[1]
CH[0]
PS_E2SCR_CHn[ACMD]
ACMD[4]
ACMD[3]
ACMD[2]
ACMD[1]
ACMD[0]
CH[2]
CH[1]
CH[0]
PS DDSR CHn
DDSR Shift Trigger
1
0

- The logic for processing the DDSR contents is shared among all the seven channels. It should be ensured by the software to correctly program the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] register contents so that the none of the bits in the command get missed. Also, since (1) is possible, it is up to the application to ensure that the UART TX_BUFFER does not remain busy to the extent that some particular DDSR bit does not get the chance to be shifted out before it gets overwritten by some new DDSR bit.
- The approximate maximum time, beginning from when PS_E2SCR_CHn[DDSR_RDY] is made "0", to the time when the command is available for transmission is less than 1us, provided no other channel has an active command processing ongoing.If commands are written to multiple DDSR registers one after the other, then the time after which the particular channel will again get an access will increase. Note that the irrespective of the sequence in which the PS_E2SCR_CHn[DDSR_RDY] of the channels are cleared, the channels are always searched in a round robin fashion for the status of the respective DDSR_RDY bits. When the internal sequencer find the respective DDSR_RDY as "0" it starts the processing of the corresponding DDSR. The sequencer always goes as Channel1->Channel2->Channel3…Channel7->Channel1…
- The logic that shifts each bit of the DDSRs of each channel is shared among all the channels. When the shift trigger of a particular DDSR is received and if the UART TX Buffer is empty then the particular command bit is transmitted to the UART TX by the appropriate CMD/ACMD. It is possible that at one point of time, CMD/ACMD of more than one channel are pending to be transmitted to the UART TX. In this case the channel with a higher channel number will be given a priority. Thus if channel2 and channel1, both have a pending CMD/ACMD shift, then channel2 would be the first to get an access. Each pending channel would queue up for transmission. It has to be ensured by the software that the internal shift triggers generated through the PS_DDTRIG_PERR_CHn and PS_DDTRIG_OFFR_CHn registers are correctly spaced so that all the channels get an access to transmit their DDSR bits. If a command bit that is desired to be shifted encounters more than one shift triggers (because it did not get shifted on the previous shift trigger) then there is no indication for this loss of synchronization. In general, if the if PS_DDTRIG_PERR is programmed to be greater than 500us(which is a practical PSI5 sync pulse period) then it would be ensured that the bits are correctly transmitted even if all the channels simultaneously transmit their command bits.
- If a command bit that is desired to be shifted, encounters more than one shift triggers (because it did not get shifted on the previous shift trigger), then the PS_E2SSR_CHn[SYNCHRO_OVF] bit gets set, to indicate synchronization

overflow. An interrupt can also be generated if the PS_E2SCR_CHn[SYNCHRO_OVF_IE] bit is set to 1. In general, if PS_DDTRIG_PERR is programmed to be greater than 500 us (which is a practical PSI5 sync pulse period), then it would be ensured that the bits are correctly transmitted even if all the channels simultaneously transmit their command bits.

57.5.4 TimeStamp

The IP provides for two global 24-bit timestamp counters. Each can be clocked by either the module clock or the GTM trigger. The two timestamp counters are detailed as TimeStamp Counter A and B. Please see Figure 57-16. Following are the details:

1. Each channel has an internal TimeStamp buffer that can be configured to capture either the TS counter A or B when it encounters the "capture event". Provided there is no XCRC error in the message, then this Timestamp buffer is appended to the message when it is written to MRU buf1. When there is an XCRC error in the message, then Channel0 timestamp is appended. Please see point number (7) below.
2. The "captured event" for the timestamp buffer of each channel can either be when a "a valid message header is detected" or when "a Command (or Alternate Command) is moved from the UART Tx Buffer to the UART Tx shift register" corresponding to the channel for which this command is being shifted to the UART.
3. The TimeStamp Buffer can be enabled or disabled through the PS_MSGA_CHn[TSBUF_EN] bit. When the TimeStamp buffer is disabled then it no longer responds to "capture events" and always contains the value "0".
4. It is possible to reset each TimeStamp buffer by using the PS_MSGA_CHn[TSBUF_CLR] bit. As soon as this bit is written to "1" the TimeStamp buffer is reset to all "0s" and it remains in this state till it encounters a "capture event".
5. In the case when the reception of message is chosen as the trigger, the time stamp will be captured immediately after the header byte of the UART message packet has been received.
6. In the case when the move of CMD or ACMD from the internal UART Tx Buffer to the UART Tx shift register in the UART module is chosen as the trigger, the timestamp for the intended channel (indicated in the CIDn fields of CMD or ACMD) is valid for all frames received on a channel between moves of CMD or ACMD from the UART Tx Buffer to the UART Tx Register (i.e., for a PSI5 SYNC period).

7. When a XCRC error is detected in any message then the timestamp that is appended with the message is the Channel0 timestamp which is captured at the time when XCRC_ERR has been detected.
8. For "illegal messages", the timestamp value that is captured is the Channel0 Timestamp value, which is captured whenever "a valid message header for that illegal message is detected". Note that the definition of a "legal" or a "illegal" message applies to a PSI5 message only when the concerned UART packet has no XCRC error.
9. In addition to the timestamp buffers described in point number (1) above, there is a separate timestamp capture that occurs in case of a watchdog error. This timestamp is captured in the PS_WDGTSSR[WDGTS_STATUS] bits. The TimeStamp A or B (as programmed in the PS_MSGA_CHn[TIMESTAMP_A_B_SEL]) is captured as soon as the watchdog overrun error occurs for that channel. Please see Watchdog Error Status and Watchdog Timestamp status register (PS_WDGTSSR), for more details.

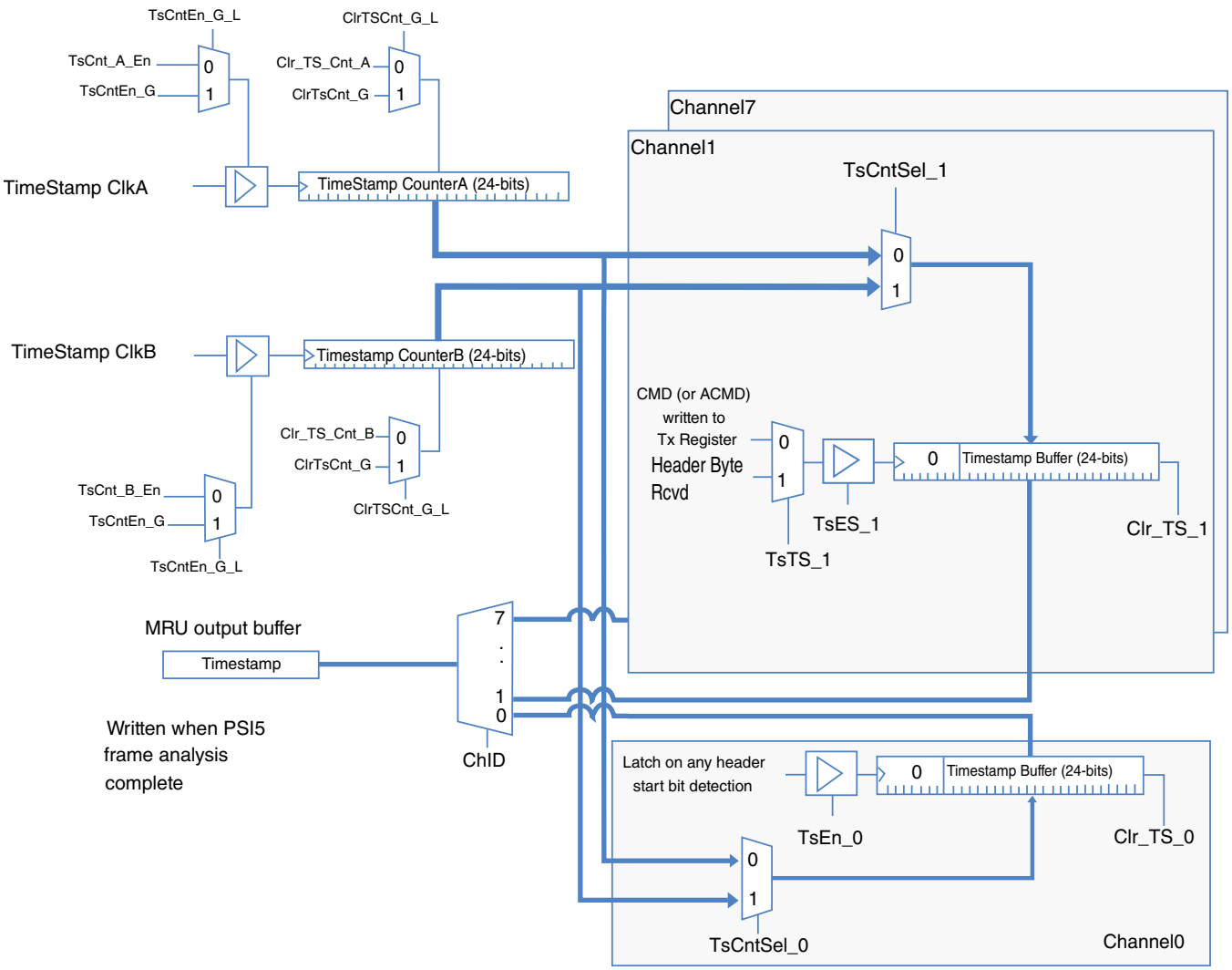


Figure 57-16. TimeStamp Capture

57.5.5 PSI5-S Watchdog Operation

The IP provides a watchdog to keep track of a number of reception parameters. The watchdog counter is a 24 bit down counter. Each channel, except Channel0, contains a watchdog counter. The refresh event for the watchdog is automatically chosen depending on whether the channels mode is synchronous or asynchronous. The watchdog timeout value can be configured in the "WD_TO" bits in the PS_WD_CFGR_CH[n] register. The refresh event for the watchdog are different in the synchronous and asynchronous modes as mentioned below:

- In Asynchronous mode:

As soon as the IP enters the PS_NORMAL mode the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value, and in case the watchdog is enabled then the counter starts immediately. Whether the WD has to be “refreshed(re-loaded)” or “not to be refreshed”, is analyzed when the XCRC is calculated. In case the XCRC is correct, the watchdog is re-loaded (It is not STOPPED) else the WD counter keeps on running normally, eventually leading to a watchdog overrun error. In case of watchdog overrun error, the counter is refreshed.

- In Synchronous Mode:

  As soon as the IP enters the PS_NORMAL mode, the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value and in case the watchdog is enabled, then the counter starts immediately. Thereby, it is re-loaded, either on the occurence of a sync pulse or the watchdog overrun error. In case the event for this load is the Sync Pulse, then it immediately starts counting again and keeps on counting till it is stopped when the “Packet_counter” reaches the count of “total number of configured slots”. The “Packet_counter” here is a counter that keeps track of the number of configured (enabled) slots. It is incremented on the reception of each UART packet with a “CID” and “FID” such that PS_MSGA_CHCID[FID_byte] is not equal to “0”. The packet counter is however incremented ONLY if the XCRC of the received UART packet is correct else it is not incremented. In case the “Packet Counter” does not reach the “expected packet count” within the “WD timeout configured”, the WD overrun error is generated. In case of a WD overrun error, the counter is stopped and does not start till a Sync Pulse is encountered.

In both the above modes, the watchdog is always refreshed on the occurrence of Watchdog overrun error (F_WD_ERR) or a software reload. Also please refer to PSI5-S channel watchdog configuration register (PSI5S_WD_CFGR_CH*n*).

The watchdog can be started in the NORMAL mode by writing a "1" to the WDEN bit in the PSI5_WD_CFGR_CH[n]. If the watchdog counter is not refreshed by its event and the watchdog counter becomes equal to "0"; the F_WD_ERR flag is set in the PSI5_IRQ_ERR_SR register and the same is also reflected in the PS_MRU_BUF2_REG1 register. An interrupt can also be generated on any of the "ipi_ps_irq[irq_n]" lines depending on the configuration. Note that writing a "0" to WDEN bit disables the watchdog checking mechanism.Figure 57-17 shows the watchdog counter.

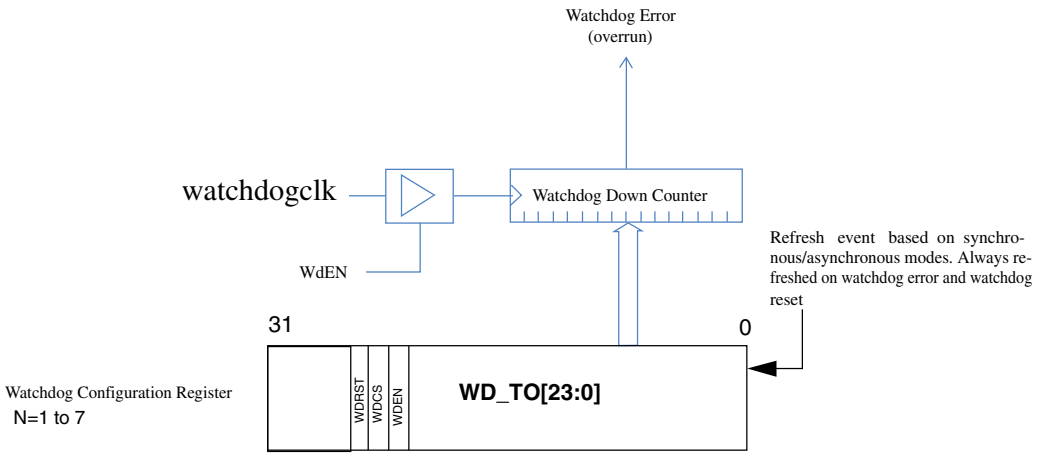


Figure 57-17. Watchdog Counter

As soon as the watchdog overrun error for a particular channel occurs, the status of the watchdog error is immediately latched in the register described in PSI5-S Watchdog Error Status and Watchdog Timestamp status register (PSI5S_WDGTSSR). This watchdog error status(PS_WDGTSSR[F_WD_ERR_STATUS[CHn]]) is also sent to the MRU registers (as F_WD_ERR) as shown in the PSI5 frame status word (shown in Figure 57-7 and Figure 57-8). However this information is sent in the next upcoming PSI5 message packet that occurs for that particular channel, provided this upcoming PSI5 message has no XCRC error. In case the upcoming UART packet has an XCRC error then the Frame watchdog error field in this PSI5 message packet is the "OR" of the F_WD_ERR status of all the channels i.e. OR of all the fields of PS_WDGTSSR[F_WD_ERR_STATUS] bits.

57.5.6 Interrupt handling

Following are the various interrupt signals that represent the various events:

1. PSI5-S general interrupt request (**Status flags in PS_MBOX_SR_IRQ and PS_ERR_SR_IRQ**)/ipi_ps_irq[irq_n] where irq_n = 0 to 7 and denotes the irq number.

   There are 8 such interrupts in the IP. Each of these "irq" lines can be configured to respond to any or all of the following events:

   - At least one message of the specific channel has been read by the DMA from the MRU buffer. (The interrupt is configurable to respond to any channel through the corresponding PS_MBOX_SEL_IRQ[irq_n] register)

- When there is a XCRC error in a new packet that is transferred to MRU buffer2 OR
- When there is a CRC or parity error in a new packet that is transferred to MRU buffer2 OR
- When the watchdog Error has been detected in the message just written to MRU buf2 OR
- When the message header has been detected with transceiver error flags OR
- When the PSI5 state machine detects in the UART, the message parity error, the overflow error or the framing error and the corresponding frame is transferred to MRU buffer 2.
- When the "R_OVL_ERR" error occurs in the new packet that is transferred to MRU buffer 2 OR
- When the "R_UVL_ERR" error occurs in the new packet that is transferred to MRU buffer 2 OR
- When the number of bytes received does not match the expected Fn bytes (N_ERR == 1)

  Each of the "ipi_ps_irq[irq_n]" lines can be configured to be asserted on any of the above events by correctly programming the registers PS_MBOX_SEL_IRQ[irq_n] and PS_ERR_SEL_IRQ[irq_n].

  This interrupt is cleared when the corresponding bits in PS_MBOX_SR_IRQ and PS_ERR_SR_IRQ are cleared. It is assumed that one event would not be mapped on more than one "irq" lines. If the same event is mapped on more than one "irq" lines then clearing the corresponding status bit in the PS_MBOX_SR_IRQ or PS_ERR_SR_IRQ will clear all the interrupt requests related to that same event.Figure 57-18 shows the interrupt structure for IRQ interrupts.

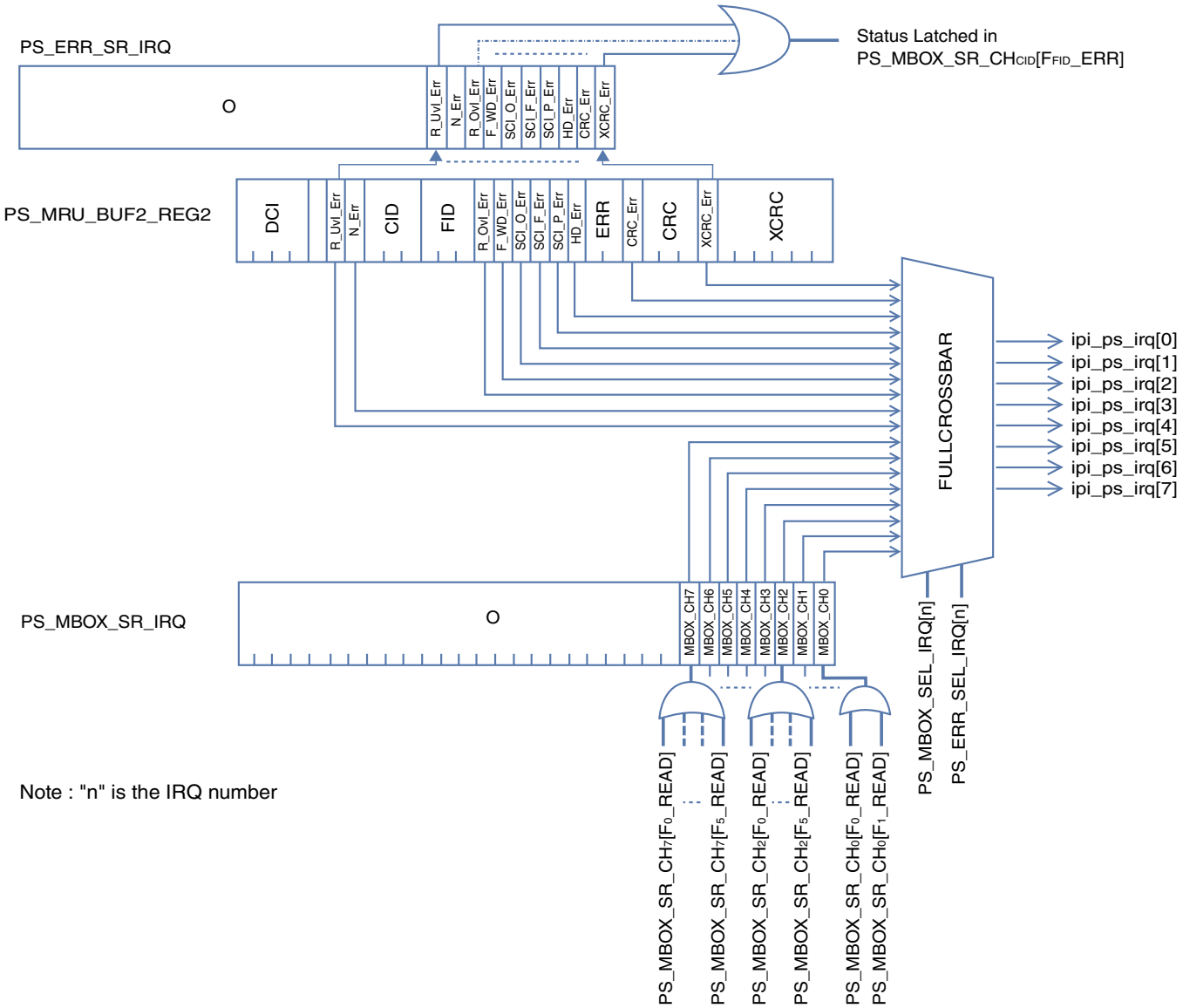


Figure 57-18. IRQ[n] interrupts

2. PSI5-S ECU-to-Sensor interrupt **(Status Flags in PSI5_E2SSR)**/ipi_ps_e2s_ch[n] where n = 1 to 7:

   This interrupt is used for the ECU to Sensor Communications and is generated in the following cases:

   - When the DDSR is empty and is ready to accept a new command new from the application. In this case the PS_E2SSR[DDSR_RDY] bit is set to "1".
   - When the DDSR is still busy (PS_E2SSR[DDSR_RDY] bit is set to "0") and the application attempts to write a new command to the DDSR. The new command is rejected and the PS_E2SSR[CMD_NWRT] bit is set to "1".
   - When the command that is written to the DDSR has finished calculating the CRC/stuff bits/start bits and is ready for transmission the

PS_E2SSR[CMDPR_BZY] goes to "1" leading to the generation of this interrupt.

- When there is a synchronisation overflow in a particular channel (the same data bit encounters 2 sync pulses without getting a chance to shift), then the bit PS_E2SSR_CHn[SYNCHRO_OVF] is set to “1”.

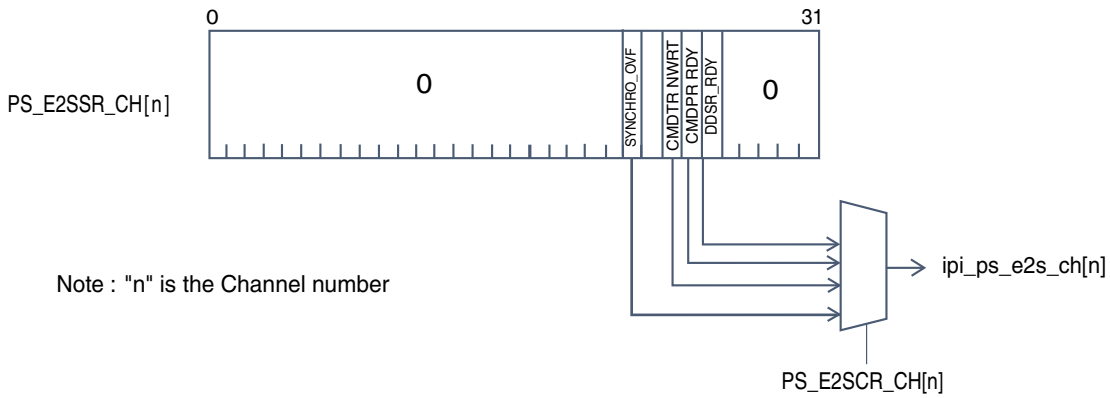


Figure 57-19. ECU to Sensor interrupts

3. Global interrupt (**Status Flags in PS_GLSR**) /ipi_ps_glbl:

   This interrupt is a single interrupt for the whole IP and is generated in the following cases.

   - When the DMA has not completed reading the MRU buffer2; simultaneously there is a pending data in MRU buffer1 that needs to be offloaded to the MRU buffer2 and a new message comes, that overwrites the contents of MRU buffer 1. The channel ID and the Frame ID of the overwritten message can be read from the PS_GLSR registers.
   - When there is any change in the "PSI5-S" global modes of the IP. Thus it is asserted during the following transitions UART_STDALONE to PS_DISABLE, PS_DISABLE to PS_CONFIG, PS_DISABLE to PS_NORMAL and PS_CONFIG to PS_NORMAL. Assertion of this interrupt indicates that the desired mode has been accomplished by the IP.
   - When the “DIRCMD_RDY” bit in the PS_GLSR is set to “1”, indicating that PS_DIRCMD register is ready to accept a new Direct Command write from the CPU.

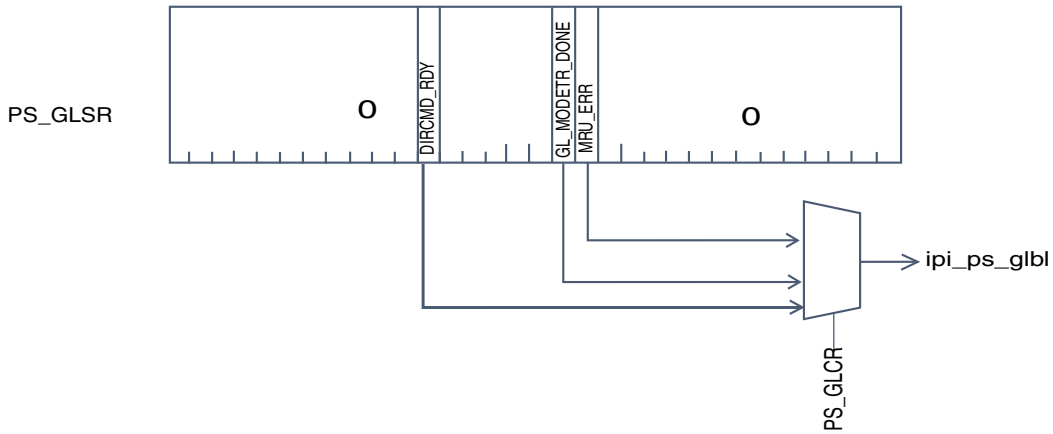


Figure 57-20. Global Interrupts

57.6 PSI5-S Clock and Reset

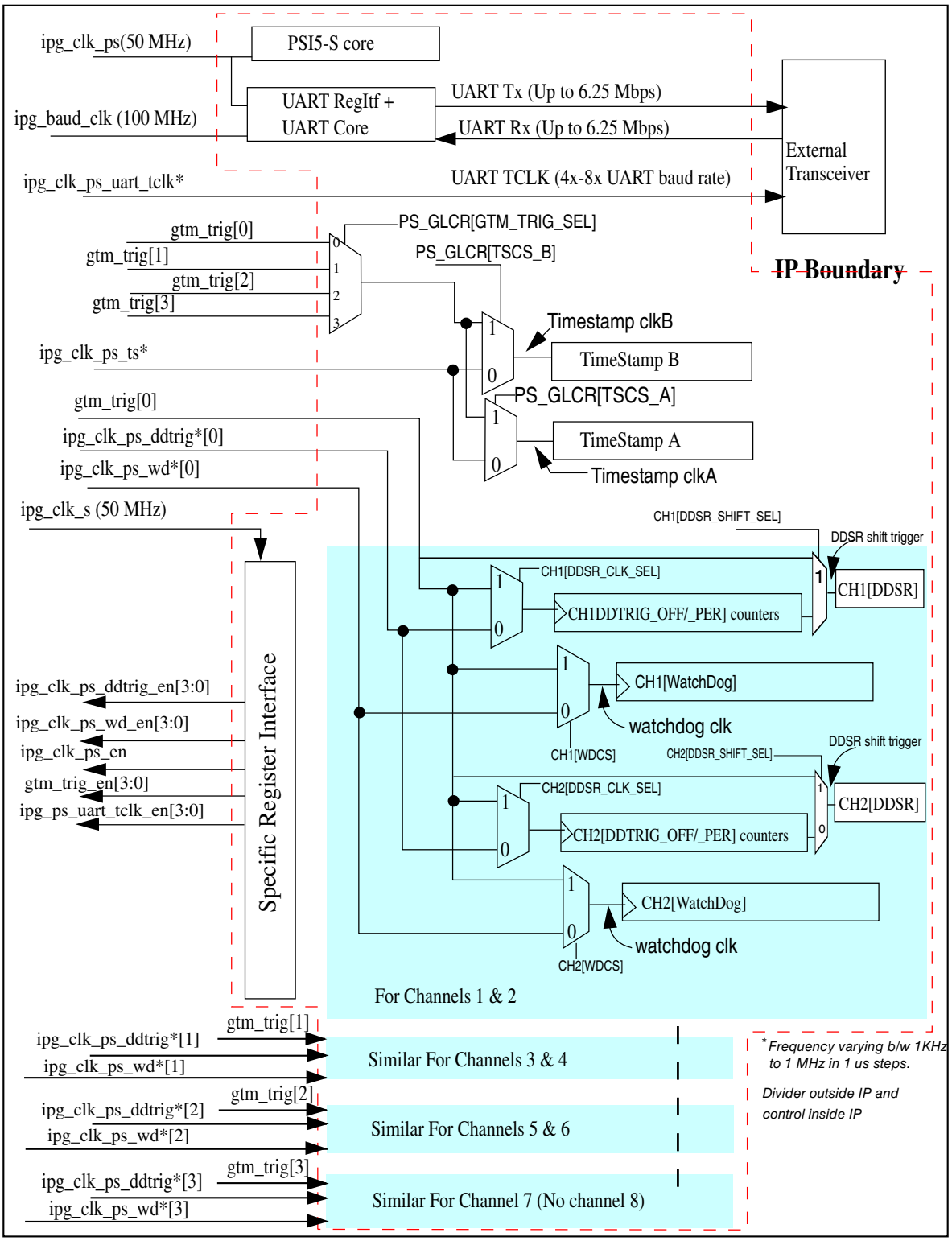


Figure 57-21. Clock distribution in PSI5-S

Figure 57-21 shows the clocking scheme for the PSI5-S IP. It also shows the distribution of various clocks on the basis of channels.

The divider and the control of clock "ipg_clk_ps_ts" is outside the IP, while for the clocks "ipg_clk_ps_wd","ipg_clk_ps_ddtrig" and "ipg_clk_ps_uart_tclk", though the frequency divider is placed outside the IP yet the value of this divider is controlled from inside IP using the "*_clkdivN"signals which are in turn driven by IPS registers of the IP.

For the safe operation of the UART module, the following relation should be satisfied between the "ipg_baud_clk" and "ipg_clk_ps":

ipg_baud_clk > *ipg_clk_ps* > ipg_baud_clk/3

Chapter 58
SENT Receiver (SRX)

58.1 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled "SENT - Single Edge Nibble Transmission for Automotive Applications" and released on January 27, 2010 (http://www.sae.org), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

58.1.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See the device configuration details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA

    - Programmable option per channel to support reading of messages via DMA or Interrupt
    - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
    - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)
    - Separate buffers for storing Fast and Slow Serial Messages
    - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel
    - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel
- Supports all bit rates by having per channel configurable Rx clock periods from 3 µs to 90 µs
- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (i.e. without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.

- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See Clocks and resets for more details on the clocks.

58.1.2 Modes of operation

Modes of operation.

58.1.2.1 Debug mode

In debug mode, message reception will be halted (without waiting for the current message being received to be completed) and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

58.1.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See System bus clock requirements) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See High frequency receiver clock (protocol clock) requirements) for accurate receive operations.

Note

Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

NOTE

If the SOC requests entry to debug or stop mode, the message being received currently is discarded and the SOC enters debug/ stop mode immediately. This does not affect the messages received completely prior to entering debug mode and will still be present on the message buffer and registers until they are read out.

NOTE

The message read registers [Channel ‘n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA), Channel ‘n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC), Channel ‘n’ Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS), Channel ‘n’ Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3), Channel ‘n’ Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2), Channel ‘n’ Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS), DMA Fast Message Data Read Register (DMA_FMSG_DATA), DMA Fast Message CRC Read Register (DMA_FMSG_CRC), DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS), DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2) and DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)] will appear to lose their contents in the following conditions:

- The very first message is being received but not yet completely received and the MCU enters debug or freeze mode, the current message reception will get discarded and message read registers (as mentioned above) will read zeros
- Auto clear functionality is enabled (GBL_CTRL[FAST_CLR] = 1). In this case, when first message is read, it will get clear the message read registers (due to auto clear functionality being enabled). On reading again, the message read registers (as mentioned above) might read zeros.

If the MCU requests entry to debug or stop mode, the message being received currently by SENT Receiver is discarded and the MCU enters debug/stop mode immediately. This does not affect the messages received completely, prior to entering debug mode and these messages will still be present on the message buffer and registers until they are read out. Thus allow one message to be received completely and do not enable "Auto Clear" (GBL_CTRL[FAST_CLR] = 0), to allow messages to be read in debug or stop mode.

58.1.3 Block diagram

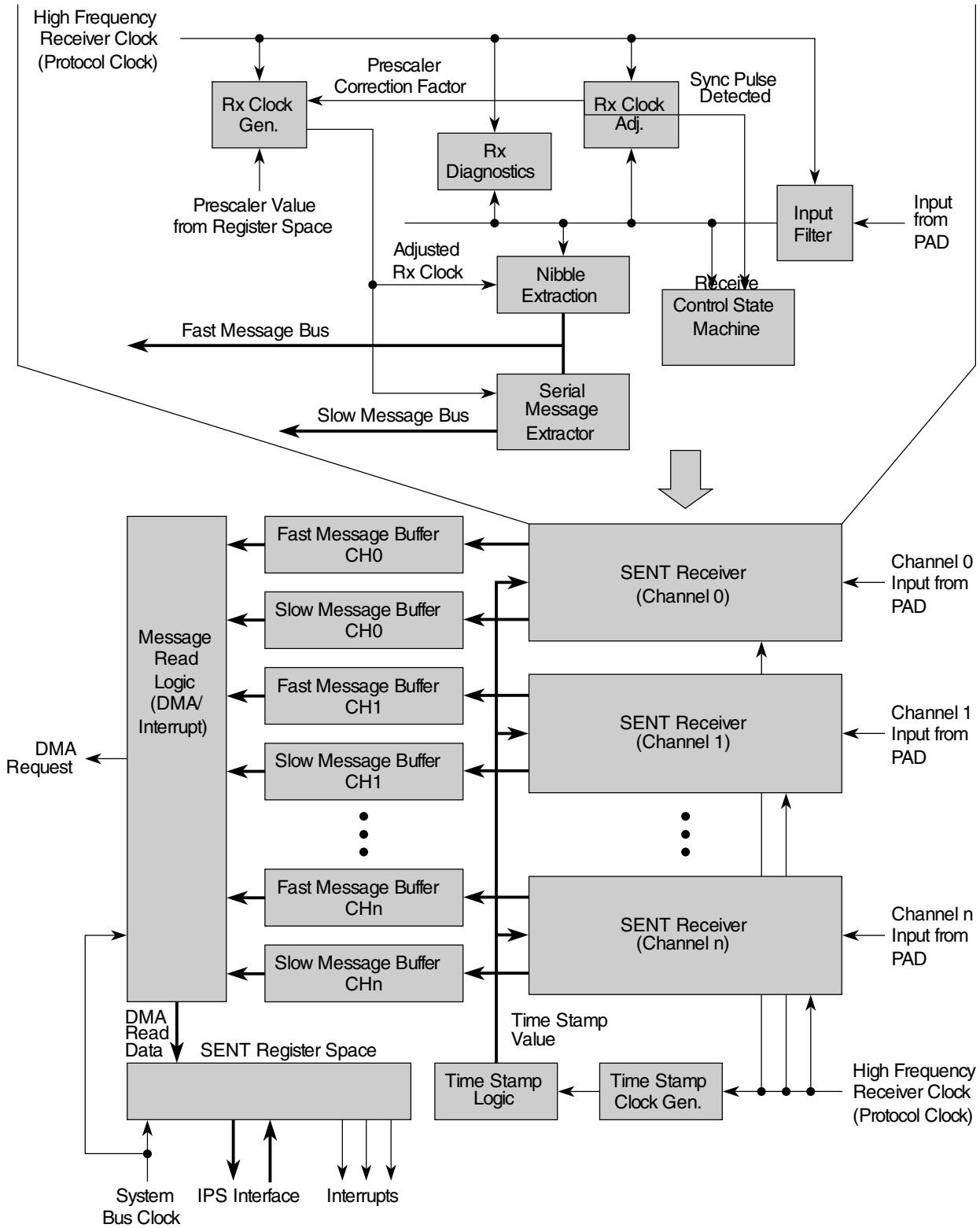


Figure 58-1. SENT Receiver block diagram

58.1.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. Figure 58-1 shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages.A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

58.1.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the message buffer as described in the subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel's status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the buffers on high frequency receiver clock. The module's message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see Functional description.

58.2 External signal description

This section lists all inputs to the SENT Receiver block.

58.2.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT Receiver channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note

The value of CH is device-specific. See the device configuration details.

58.3 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT Receiver block.

The following table gives an overview on all SENT Receiver registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the device configuration section for details.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

SRX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Control Register (SRX_GBL_CTRL)	32	R/W	0000_0010h	58.3.1/3092
4	Channel Enable Register (SRX_CHNL_EN)	32	R/W	0000_0000h	58.3.2/3093
8	Global Status Register (SRX_GBL_STATUS)	32	w1c	0000_0000h	58.3.3/3094
C	Fast Message Ready Status Register (SRX_FMSG_RDY)	32	R/W	0000_0000h	58.3.4/3095
10	Slow Serial Message Ready Status Register (SRX_SMSG_RDY)	32	R/W	0000_0000h	58.3.5/3096
18	Data Control Register 1 (SRX_DATA_CTRL1)	32	R/W	1111_1111h	58.3.6/3097

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	Fast Message DMA Control Register (SRX_FDMA_CTRL)	32	R/W	0000_0000h	58.3.7/3099
2C	Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)	32	R/W	0000_0000h	58.3.8/3100
34	Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)	32	R/W	0000_0000h	58.3.9/3100
38	Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)	32	R/W	0000_0000h	58.3.10/ 3101
40	DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)	32	R	0000_0000h	58.3.11/ 3102
44	DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)	32	R	0000_0000h	58.3.12/ 3103
48	DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)	32	R	0000_0000h	58.3.13/ 3103
50	DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)	32	R	0000_0000h	58.3.14/ 3104
54	DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)	32	R	0000_0000h	58.3.15/ 3105
58	DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)	32	R	0000_0000h	58.3.16/ 3106
60	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH0_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
64	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH0_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
68	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH0_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
70	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH1_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
74	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH1_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
78	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH1_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
80	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH2_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
84	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH2_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
88	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH2_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
90	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH3_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
94	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH3_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
98	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH3_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
A0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH4_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH4_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
A8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH4_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
B0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH5_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
B4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH5_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
B8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH5_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
C0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH6_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
C4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH6_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
C8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH6_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
D0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH7_CLK_CTRL)	32	R/W	0000_8000h	58.3.17/ 3106
D4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH7_STATUS)	32	w1c	0000_0000h	58.3.18/ 3108
D8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH7_CONFIG)	32	R/W	0000_0104h	58.3.19/ 3111
160	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
164	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
168	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
16C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH0_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
170	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH0_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
174	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
178	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
17C	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
180	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
184	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH1_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
188	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH1_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18C	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
190	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
194	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
198	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
19C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH2_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
1A0	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH2_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
1A4	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH2_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
1A8	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
1AC	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
1B0	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
1B4	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH3_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
1B8	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH3_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
1BC	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH3_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
1C0	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
1C4	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
1C8	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
1CC	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH4_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
1D0	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH4_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
1D4	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH4_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
1D8	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
1DC	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
1E0	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116

Table continues on the next page...

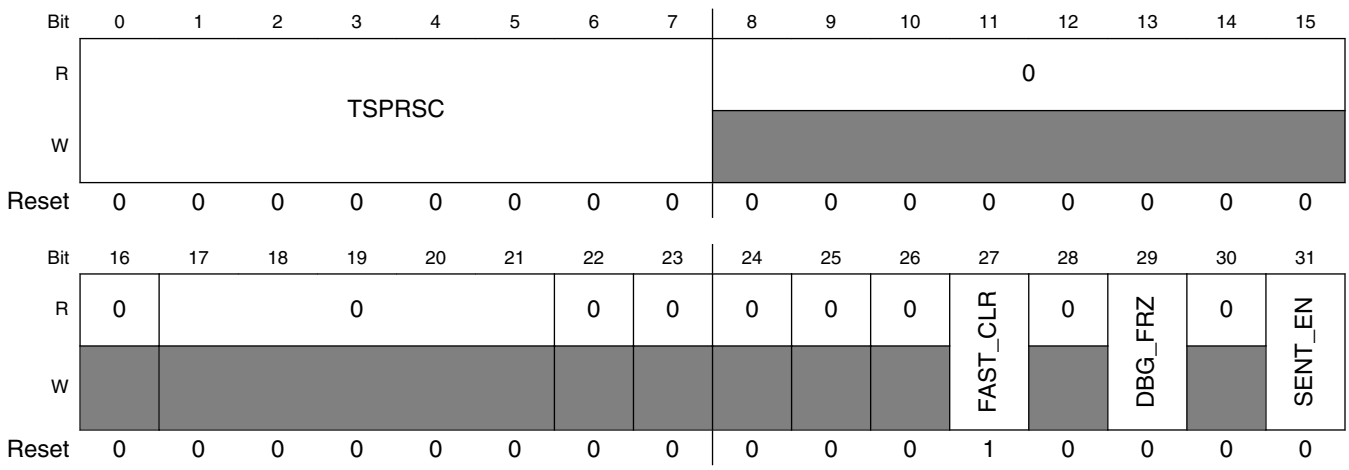
SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E4	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH5_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
1E8	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH5_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
1EC	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH5_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
1F0	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH6_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
1F4	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH6_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
1F8	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH6_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
1FC	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH6_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
200	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH6_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
204	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH6_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118
208	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH7_FMSG_DATA)	32	R	0000_0000h	58.3.20/ 3114
20C	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH7_FMSG_CRC)	32	R	0000_0000h	58.3.21/ 3115
210	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH7_FMSG_TS)	32	R	0000_0000h	58.3.22/ 3116
214	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH7_SMSG_BIT3)	32	R	0000_0000h	58.3.23/ 3116
218	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH7_SMSG_BIT2)	32	R	0000_0000h	58.3.24/ 3117
21C	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH7_SMSG_TS)	32	R	0000_0000h	58.3.25/ 3118

58.3.1 Global Control Register (SRX_GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Address: 0h base + 0h offset = 0h



SRX_GBL_CTRL field descriptions

Field	Description
0–7 TSPRSC	Time Stamp Prescaler Value. Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. 1 to 255 - Time Stamp Clock = (High Frequency Clock)/(Value + 1). Value set by user software should be such that the clock period is at least 1 µs. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4A (or 74) to obtain a 1 µs clock. Note: Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_GBL_CTRL field descriptions (continued)

Field	Description
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FAST_CLR	Fast Clearing Enable bit. Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. SeeFast Message Ready Status Register (SRX_FMSG_RDY) andSlow Serial Message Ready Status Register (SRX_SMSG_RDY) for details on the ready status bits. 0 Fast Clearing is disabled. Bits will be cleared by writing 1 1 Fast Clearing is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DBG_FRZ	Debug Freeze. This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 No effect in debug mode 1 Freeze in debug mode
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SENT_EN	SENT Receiver Global Enable. This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 Entire module is disabled 1 Module is enabled

58.3.2 Channel Enable Register (SRX_CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

NOTE

Reads of bits beyond those for the supported number of channels should be ignored, and these bits must be written to 0.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																EN_CH															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CHNL_EN field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 EN_CH	Enable bits for channels 7 to 0. All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 Channel is disabled 1 Channel is enabled. Channel parameter registers are locked for writing

58.3.3 Global Status Register (SRX_GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						FMDU	SMDU	0							
W							w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_GBL_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FMDU	Fast Message DMA Underflow: Interrupt Status bit that indicates when there is an underflow in Fast Message buffers in any channel which has DMA enabled. This bit is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0.

Table continues on the next page...

SRX_GBL_STATUS field descriptions (continued)

Field	Description
	0 No underflow 1 Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow: Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 No underflow 1 Underflow occurred
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.3.4 Fast Message Ready Status Register (SRX_FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (viaFast Message DMA Control Register (SRX_FDMA_CTRL)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (viaFast Message Ready Interrupt Control Register (SRX_FRDY_IE)) and the channel is enabled, the message ready status bit in this register will be asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in theGlobal Control Register (SRX_GBL_CTRL) .

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								F_RDY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FMSG_RDY field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 F_RDY	Fast Message Data Ready Status bits for channels 7 to 0. Each bit indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **NOTE:** These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR BIT in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See SectionGlobal Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. **NOTE:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **NOTE:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Fast Message is available to be read out via Interrupt 1 Fast Message is available to be read out via Interrupt

58.3.5 Slow Serial Message Ready Status Register (SRX_SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (viaFast Message DMA Control Register (SRX_FDMA_CTRL) andSlow Serial Message DMA Control Register (SRX_SDMA_CTRL)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (viaFast Message Ready Interrupt Control Register (SRX_FRDY_IE) andSlow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)) and the channel is enabled, the message ready status bit in this register will be asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in theGlobal Control Register (SRX_GBL_CTRL).

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								S_RDY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SMSG_RDY field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 S_RDY	Slow Serial Message Data Ready Status bits for channels 7 to 0. Each bit indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **NOTE:** These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR bit in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See SectionGlobal Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. **NOTE:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **NOTE:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Slow Serial Message is available to be read out via Interrupt 1 Slow Serial Message is available to be read out via Interrupt

58.3.6 Data Control Register 1 (SRX_DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

NOTE

Reset value changes depending on the number of channels that are enabled on the device.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH0			0	NIBBCH1			0	NIBBCH2			0	NIBBCH3		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH4			0	NIBBCH5			0	NIBBCH6			0	NIBBCH7		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

SRX_DATA_CTRL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 NIBBCH0	Number of Data Nibbles supported in Channel 0 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 NIBBCH1	Number of Data Nibbles supported in Channel 1 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 NIBBCH2	Number of Data Nibbles supported in Channel 2 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 NIBBCH3	Number of Data Nibbles supported in Channel 3 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 NIBBCH4	Number of Data Nibbles supported in Channel 4 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 NIBBCH5	Number of Data Nibbles supported in Channel 5 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_DATA_CTRL1 field descriptions (continued)

Field	Description
25–27 NIBBCH6	Number of Data Nibbles supported in Channel 6 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 NIBBCH7	Number of Data Nibbles supported in Channel 7 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel

58.3.7 Fast Message DMA Control Register (SRX_FDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

RW

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								FDMA_EN							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FDMA_CTRL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 FDMA_EN	Enable DMA for Fast Messages on channels 7 to 0. These bits are writable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 DMA for Fast Messages is disabled 1 DMA for Fast Messages is enabled

58.3.8 Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								SDMA_EN							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SDMA_CTRL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SDMA_EN	Enable DMA for Slow Serial Messages on channels 7 to 0. These bits are writable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 DMA for Slow Serial Messages is disabled 1 DMA for Slow Serial Messages is enabled

58.3.9 Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								FRDY_IE							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FRDY_IE field descriptions

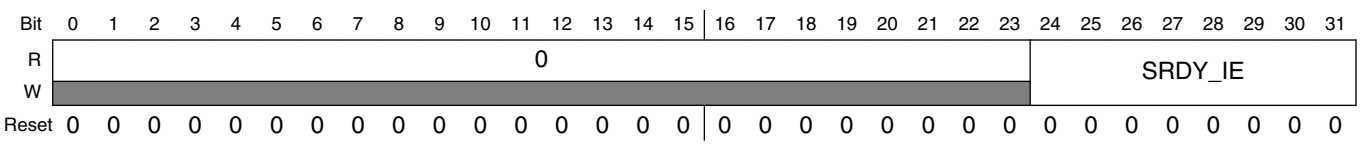
Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 FRDY_IE	Enable for Fast Message Ready Interrupt on channels 7 to 0. These bits are writable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Fast Messages is disabled 1 Interrupt on reception of Fast Messages is enabled

58.3.10 Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 38h offset = 38h



SRX_SRDY_IE field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SRDY_IE	Enable for Slow Serial Message Ready Interrupt on channels 7 to 0. These bits are writable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Slow Serial Messages is disabled 1 Interrupt on reception of Slow Serial Messages is enabled

58.3.11 DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHNUM				SCNIB				DNIB1				DNIB2				DNIB3				DNIB4				DNIB5				DNIB6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 7.
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

58.3.12 DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.3.13 DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in theGlobal Control Register (SRX_GBL_CTRL) register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

58.3.14 DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)

The above register (DMA_SMSG_BIT3) is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown inTable 58-1,Table 58-2 andTable 58-3.Table 58-1 andTable 58-2 shows the enhanced serial messages format to be used when reading this register for these two types of messages.Table 58-3 shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

NOTE

If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID and DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced Serial Message and using CFG bit, software can decode the ID and DATA fields.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 7.
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.3.15 DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown inTable 58-1,Table 58-2 andTable 58-3.Table 58-1, andTable 58-2 shows the enhanced serial messages format to be used when reading this register for these two types of messages.Table 58-3 shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0				DATA[11:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit inDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

58.3.16 DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in theGlobal Control Register (SRX_GBL_CTRL) register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

58.3.17 Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)

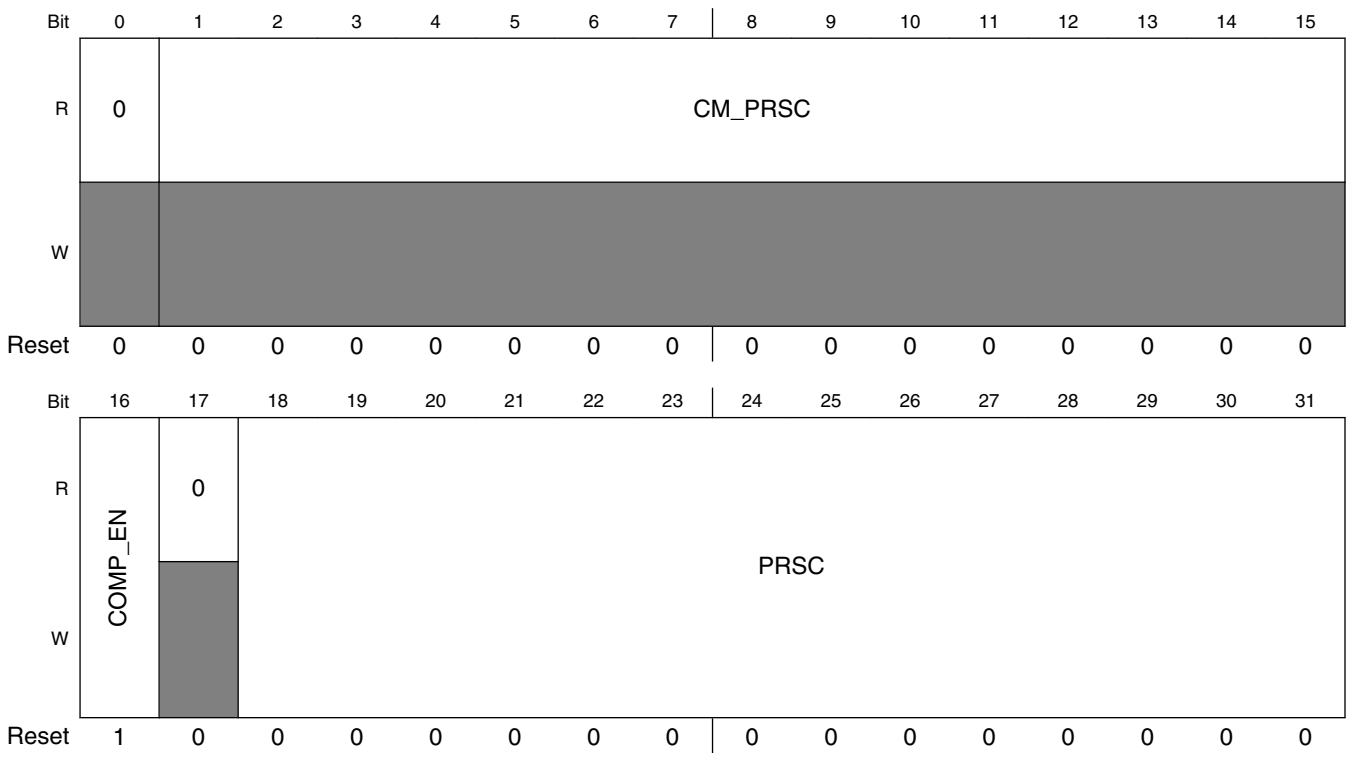
NOTE

The value of the CH is device-specific. See the device configuration section for details.

The SENT Receiver module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations and clock drift that might occur in the Sensor Tx Clock. SeeAdjustment for variation in sensor (Tx) clock for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in theChannel Enable Register (SRX_CHNL_EN) .

Address: 0h base + 60h offset + (16d × i), where i=0d to 7d



SRX_CH*n*_CLK_CTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CM_PRSC	Compensated Prescaler Value. This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as

Table continues on the next page...

SRX_CH*n*_CLK_CTRL field descriptions (continued)

Field	Description
	Compensated Prescaler Value x High Frequency Receiver Clock Period Note: The value provided in this field is ceil value of actual tick period as measured by the Receiver.
16 COMP_EN	Compensation Enable. To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 Compensation is disabled 1 Compensation is enabled
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 PRSC	Rx Prescaler Value. This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (Trx_clk) is not divisible by the high frequency receiver clock (protocol clock) period (Thf_clk), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 Prescaler is bypassed. Should not be used. Non-Zero Value Enables Prescaler The required value is computed by software as follows: $$\text{Prescaler\_value} = \text{floor}\left(\frac{T_{\text{rx\_clk}}}{T_{\text{hf\_clk}}}\right)$$ **Equation 73** (both clock periods taken in µs). Please seeHigh frequency receiver clock (protocol clock) requirements before setting this value. e.g. if the required receiver clock tick period is 43 µs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 µs) then, Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

58.3.18 Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)

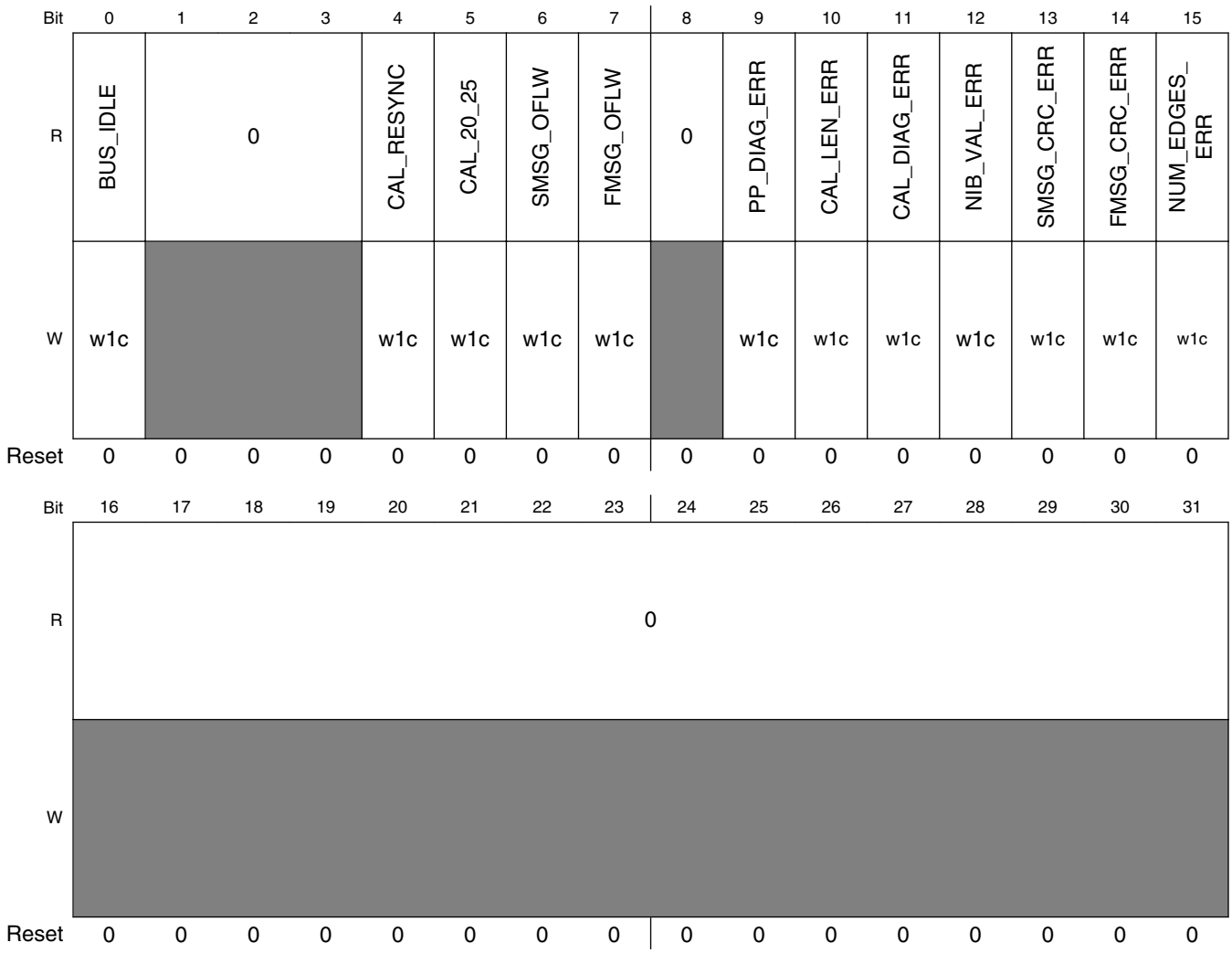
This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in theChannel Enable Register (SRX_CHNL_EN) bit for the corresponding channel.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 64h offset + (16d × i), where i=0d to 7d



SRX_CH*n*_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status. This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value(inChannel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). CPU should write 1 to this clear this bit once it is read. 0 Bus is not idle 1 Channel has been idle for more than the allowed value
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized. This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to

Table continues on the next page...

SRX_CH*n*_STATUS field descriptions (continued)

Field	Description
	assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 No interrupt 1 Interrupt Status condition has occurred
5 CAL_20_25	Calibration Variation 20 - 25% Interrupt Status. This bit indicates that the calibration pulse received on this channel has variation in between ±20% to ±25% from 56 ticks. 0 No Interrupt 1 Interrupt Status condition has occurred
6 SMSG_OFLW	Slow Serial Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow will cause state machine to halt stopping reception of new messages. 0 No Interrupt 1 Interrupt Status condition has occurred
7 FMSG_OFLW	Fast Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt 1 Interrupt Status condition has occurred
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 PP_DIAG_ERR	This diagnostic checks status bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than ±1.5625% between two messages. This check is valid only for messages with pause pulse. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
10 CAL_LEN_ERR	This diagnostic checks status bit indicates that Calibration pulse is more than 56 ticks ±25%. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
11 CAL_DIAG_ERR	Successive Calibration pulses differ by ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (SeeHigh frequency receiver clock (protocol clock) requirements). This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
12 NIB_VAL_ERR	Any nibble data value <0 or >15. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
13 SMSG_CRC_ERR	Checksum error in Slow Serial Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it.

Table continues on the next page...

SRX_CH*n*_STATUS field descriptions (continued)

Field	Description
	0 Error Check has passed 1 Error Check has failed
14 FMSG_CRC_ERR	Checksum error in Fast Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
15 NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel. This bit will not set in case of "Option 2" for successive calibration pulse check is selected. This error will only set when "Option 1" for successive calibration check is selected (SUCC_CAL_CHK). This bit is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.3.19 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained inChannel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS) and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in theChannel Enable Register (SRX_CHNL_EN) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 68h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK	FIL_CNT							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SRX_CH*n*_CONFIG field descriptions

Field	Description
0–3 BUS_IDLE_CNT	Bus Idle Count. This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled. 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts
4 IE_CAL_ RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of "Option 2" being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled 1 Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled 1 Interrupt is enabled
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 Interrupt is disabled 1 Interrupt is enabled
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
8 FCRC_CHK_ OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 Check is enabled 1 Check is disabled/off
9 IE_PP_DIAG_ ERR	Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
10 IE_CAL_LEN_ERR	Calibration pulse is wider than 56 ticks ±25%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
11 IE_CAL_DIAG_ERR	Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
12 IE_NIB_VAL_ERR	Any nibble data value <0 or >15. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
13 IE_SMSG_CRC_ERR	Checksum error in Slow Serial Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
14 IE_FMSG_CRC_ERR	Checksum error in Fast Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
15 IE_NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast Messages will be received even if data nibbles are same 1 Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable 1 25% variation is acceptable
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble is XORed)
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 Status and Communication Nibble not included in CRC calculation 1 Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble XORed)
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse. 0 Detection of Pause Pulse is disabled 1 Detection of Pause Pulse enabled
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 i.e. Low Latency Option as per SAE Specification 1 Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24–31 FIL_CNT	Input Filter Sample Count. This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0 No filtering. Input from device's pad is only synchronized 1 Non-Zero Filtering is enabled

58.3.20 Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_DATA)

Address offset: 0x0160 + 0x18 x n

This is register CHn_FMSG_DATA. The contents of this register are same asDMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 160h offset + (24d × i), where i=0d to 7d

Bit	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R	CHNUM	SCNIB	DNIB1	DNIB2	DNIB3	DNIB4	DNIB5	DNIB6
W								
Reset	0000	0000	0000	0000	0000	0000	0000	0000

SRX_CH*n*_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 7.
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

58.3.21 Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_CRC)

Address offset: 0x0164 + 0x18 x n

This is register CHn_FMSG_CRC. The contents of this register are same asDMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC) .

Address: 0h base + 164h offset + (24d × i), where i=0d to 7d

Bit	0–11	12–15	16–31
R	0	CRC4b	0
W			
Reset	0000 0000 0000	0000	0000 0000 0000 0000

SRX_CH*n*_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_CH*n*_FMSG_CRC field descriptions (continued)

Field	Description
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.3.22 Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_TS)

Address offset: 0x0168 + 0x18 x n

This is register CHn_FMSG_TS. The contents of this register are same as DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS).

Address: 0h base + 168h offset + (24d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

58.3.23 Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT3)

Address offset: 0x016C + 0x18 x n

This is register CHn_SMSG_BIT3. The contents of this register are same asDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 16Ch offset + (24d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 7.
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.3.24 Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT2)

Address offset: 0x0170 + 0x18 x n

This is register CHn_SMSG_BIT2. The contents of this register are same asDMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2) .

Address: 0h base + 170h offset + (24d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0				DATA[11:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit inDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

58.3.25 Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_SMSG_TS)

Address offset: 0x0174 + 0x18 x n

This is register CHn_SMSG_TS. The contents of this register are same as DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS).

Address: 0h base + 174h offset + (24d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

58.4 Functional description

This section describes the function of the SRX.

58.4.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the Channel Enable Register (SRX_CHNL_EN). There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in Global Control Register (SRX_GBL_CTRL) to generate the time stamping clock
- Rx Prescaler Value in Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) togenerate the channel's receiver clock
- Compensation enable in Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)
- Channel Configuration settings in the Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)
- Number of Data Nibbles in Channel in the Data Control Register 1 (SRX_DATA_CTRL1)
- Control bits that control whether messages received in channel are to read by interrupt or DMA in Fast Message DMA Control Register (SRX_FDMA_CTRL), Slow Serial Message DMA Control Register (SRX_SDMA_CTRL), Fast Message Ready Interrupt Control Register (SRX_FRDY_IE) and Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

58.4.2 DMA read logic

The SENT Receiver module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in Time stamp logic.

58.4.2.1 DMA request for Fast Message reading

Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note the DMA Controller must read all 3 DMA Read Registers to obtain the complete message. The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
  - DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)
  - DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)
  - DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)
- A fixed round robin sequencing runs, as shown in Figure 58-2, and this logic checks for messages in sequence 0→1→3 →5→7 and back to channel 0 and so on. Channel 2 is skipped as it is not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

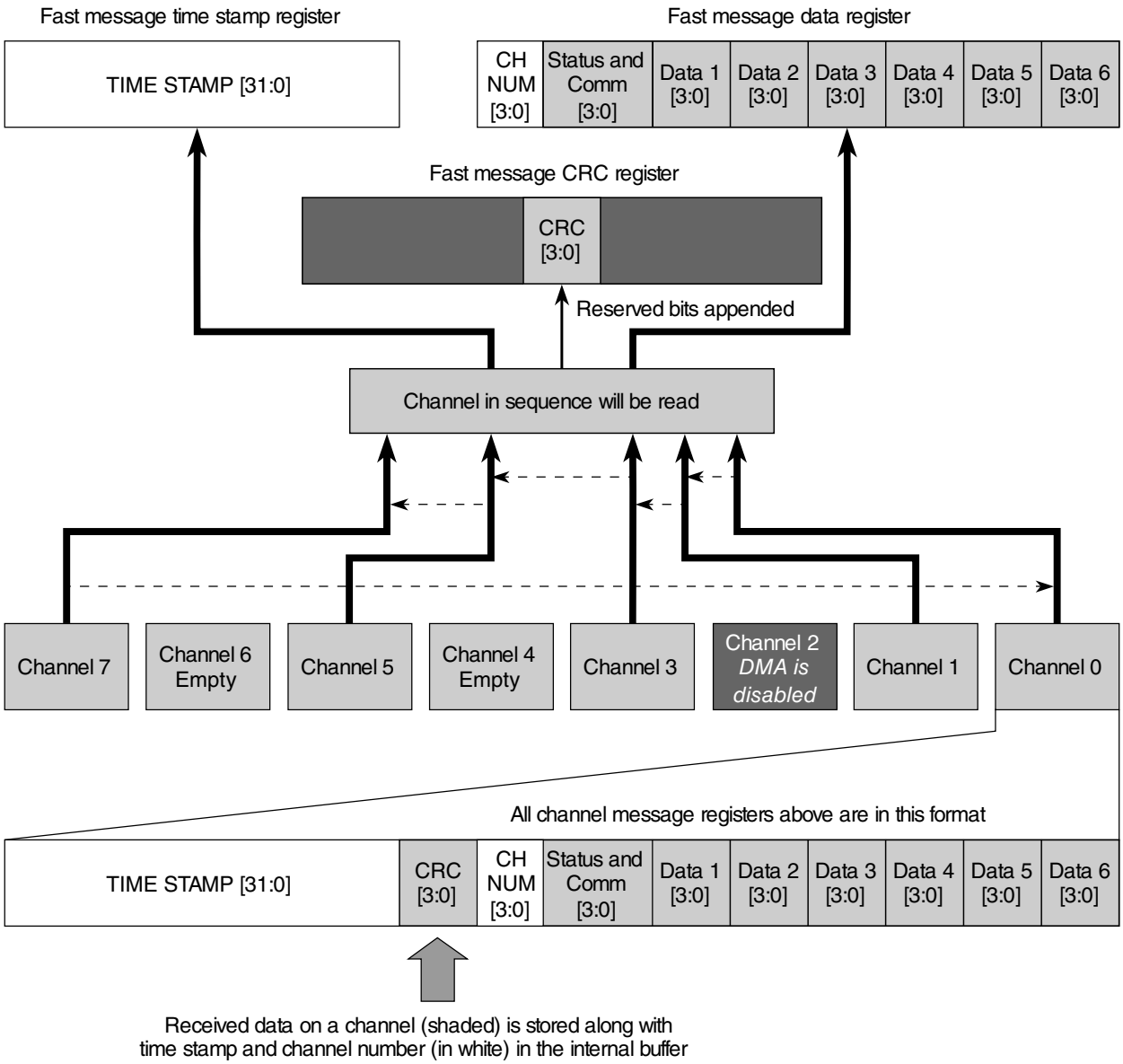


Figure 58-2. Fast Message DMA read logic

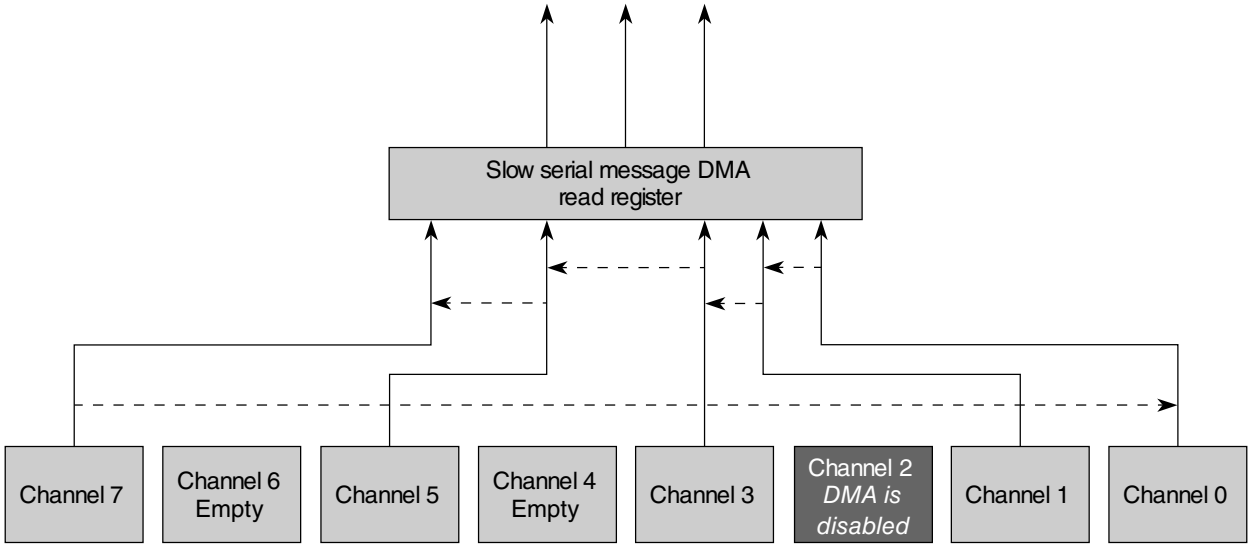
58.4.2.2 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers

- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
  - DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)
  - DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)
  - DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 DMA read accesses to the above mentioned registers)
- The channel number and time stamp will also be appended for every message
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- Since serial messages receive rate is very slow, user software should ensure there is no overflow
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see DMA request for Fast Message reading)

Figure 58-3 shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in

Figure 58-3. Slow Serial Message DMA read logic

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in figures below.

Table 58-1. Enhanced Serial Message with 8-bit ID field

31	1		16	15	0						0
CH NUM [3:0]	P				0	C	ID [7:4]	0	ID [3:0]		0
		CRC [5:0]			Data Field [11:0]						
TIME STAMP [31:0]											

Table 58-2. Enhanced Serial Message with 4-bit ID field

31	1		16	15	1						0
CH NUM [3:0]	P				0	C	ID [3:0]	0	Data Field [15:12]		0
		CRC [5:0]			Data Field [11:0]						
TIME STAMP [31:0]											

Table 58-1 and Table 58-2 shows the enhanced serial messages. Bit C (shown in figure) or CFG (as shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Table 58-3 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

Table 58-3. Short serial message

31	0	16	15	0
CH NUM [3:0]	P			
		CRC [3:0]	ID [3:0]	Data Field [7:0]
TIME STAMP [31:0]				

Again, three 32-bit accesses to be made by DMA to get one complete message with time stamp.

58.4.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

58.4.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable

- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

58.4.4 Overflow behavior

This section describes the overflow behavior.

58.4.4.1 Fast messages buffers

Overflow in Fast Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (i.e. all 3 read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

58.4.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

58.4.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register (Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)). Software can use this value to determine the variation in Tx Clock.

Note

A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

58.4.5.1 Adjustment for nibble length variation

Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

58.4.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). A timing diagram of the input filter when FIL_CNT is 4, is shown in the following figure.

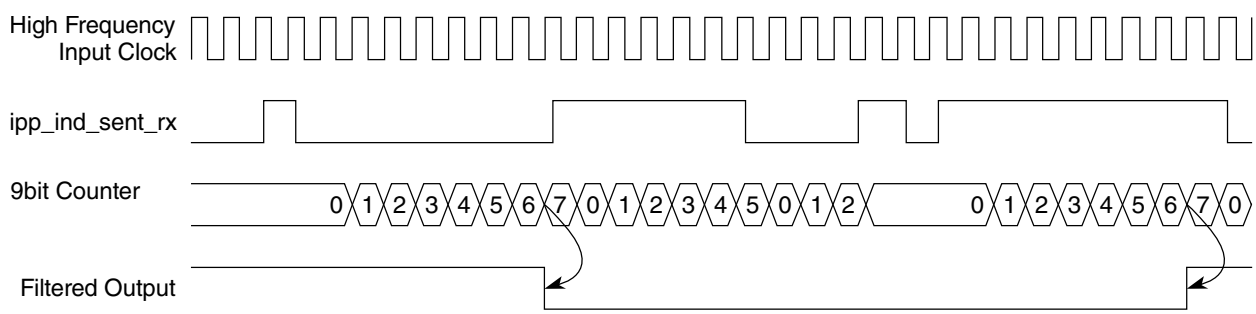


Figure 58-4. Input programmable filter timing diagram

58.4.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%

- Not the expected number of falling edges between calibration pulses. (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or Recommended.
- Any nibble data values measured as < 0 or > 15
- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

58.4.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

58.4.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%. So if protocol clock is T_{HF_CLK} and channel tick period is

T_{TX_CLK} then the check will be of 1.5625% ± {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. So it won't allow variance in clock of a channel by 1.5625% - {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

58.4.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

58.4.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 nibble values(11.5 ticks to 27.5 ticks):

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

58.4.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new recommended CRC implementation via a control bit in the channel's corresponding control register. Please refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

58.4.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic

to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel ‘n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length, the NUM_EDGES error can get asserted spuriously (Channel ‘n' Status Register - CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. Table 1 contains information due to erratum behavior and Table 2 below contains clarification of normal NUM_EDGES_ERR behavior.

Table 58-4. Behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are	Ignore both NUM_EDGES_ERR errors

Table continues on the next page...

Table 58-4. Behavior of NUM_EDGES_ERR (continued)

		seen as back to back calibration pulses and no edges in between.	
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error

Table 58-5. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

58.4.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in Global Control Register (SRX_GBL_CTRL)) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 µs. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value is 59 to generate a time stamp clock of 1 µs resolution.

Note

The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message i.e. when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

58.4.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

58.4.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in Channel 'n' Configuration Register (n=0 to (CH-1))

(SRX_CH*n*_CONFIG)). Status bit (bus_idle in Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

58.5 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

58.5.1 Clocking strategy

The following figure shows the clock domain in which each module is functioning.

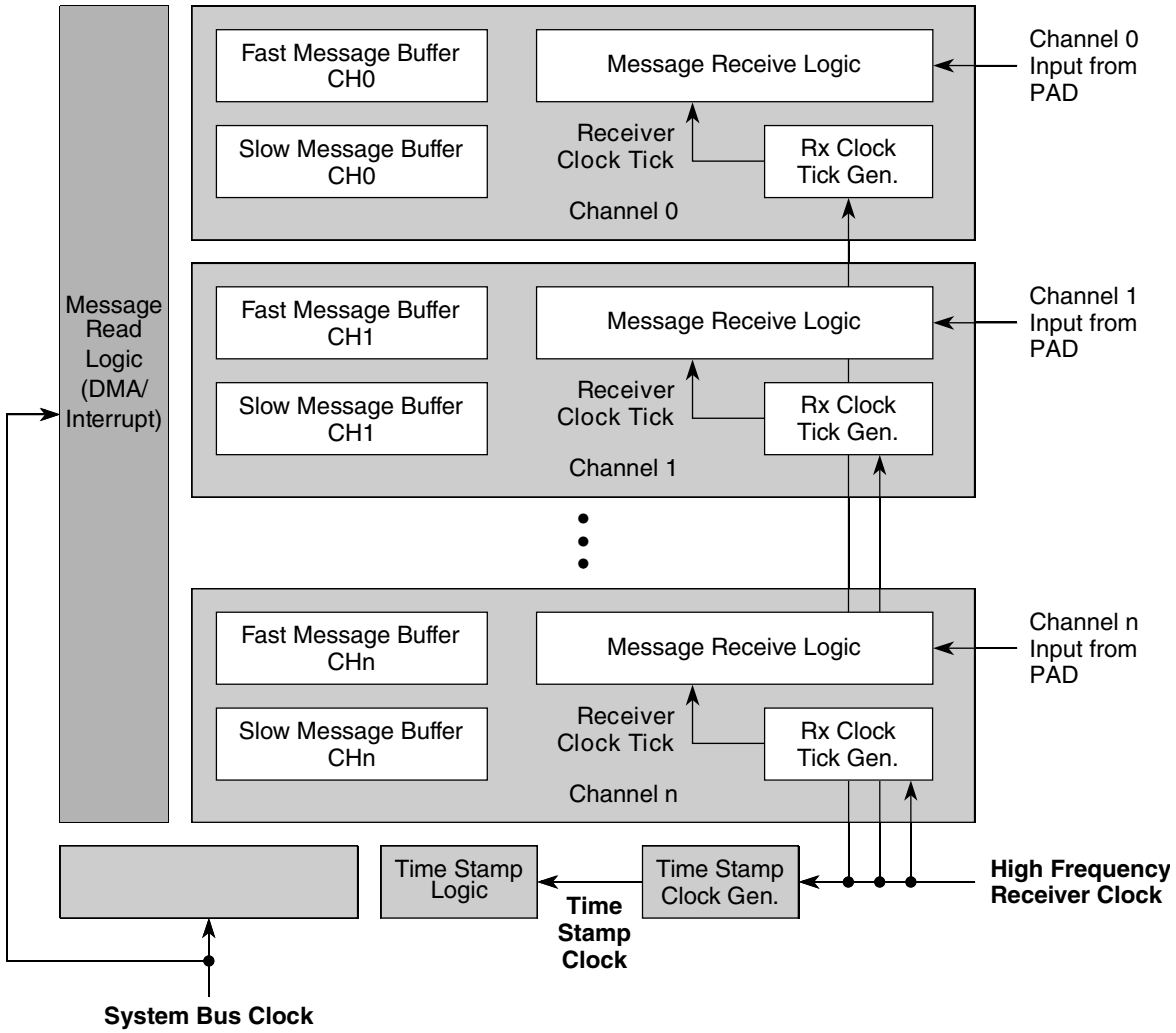


Figure 58-5. SENT Receiver module clock domains

58.5.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 µs receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 µs (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as $154 \times 2.4 = 369.6$

µs. Now since messages are arriving in parallel, 369.6 µs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that 'N_{CHNL}' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be 369.6/N_{CHNL} µs or ~370/N_{CHNL} µs. Since the CPU or the DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as 'T_{RD}' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

$$3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370 / N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

$$3 \times T_{RD} + T_{WAKUP} < 370 / N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS\_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS\_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

$$(3 \times N_{RD} + N_{ISR}) \times T_{SYS\_CLK} + T_{WAKUP} < 370 / N_{CHNL}$$

(all units is µs)

For DMA,

$$3 \times N_{RD} \times T_{SYS\_CLK} + T_{WAKUP} < 370 / N_{CHNL}$$

(all units is µs)

58.5.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula: Guard Band > 2.5 * T_{HF_CLK}

where T_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in ns and Guard Band is guard band in ns as specified in SAE specification, at particular transmitter uTick.

Note

T_{HF_CLK} in the above equation should be chosen based on the fastest sensor clock (with minimum guard band as specified in SAE spec) among all channels in the device.

Chapter 59
LINFlexD

59.1 Introduction

The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, and 2.1. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

59.1.1 Glossary and acronyms

Table 59-1. Glossary and acronyms

Term	Description
LIN	Local interconnect network
UART	Universal asynchronous transmitter receiver
ID	LIN identifier field
DMA	Direct memory access
TCD	Transfer control descriptor
IPS	Peripheral Bus Interface
IRQ	Interrupt request
FSM	Finite state machine

Table continues on the next page...

Table 59-1. Glossary and acronyms (continued)

Term	Description
WS	Wait state
SW	Software
CPU	Central processing unit
SoC	System on a chip

59.1.2 References

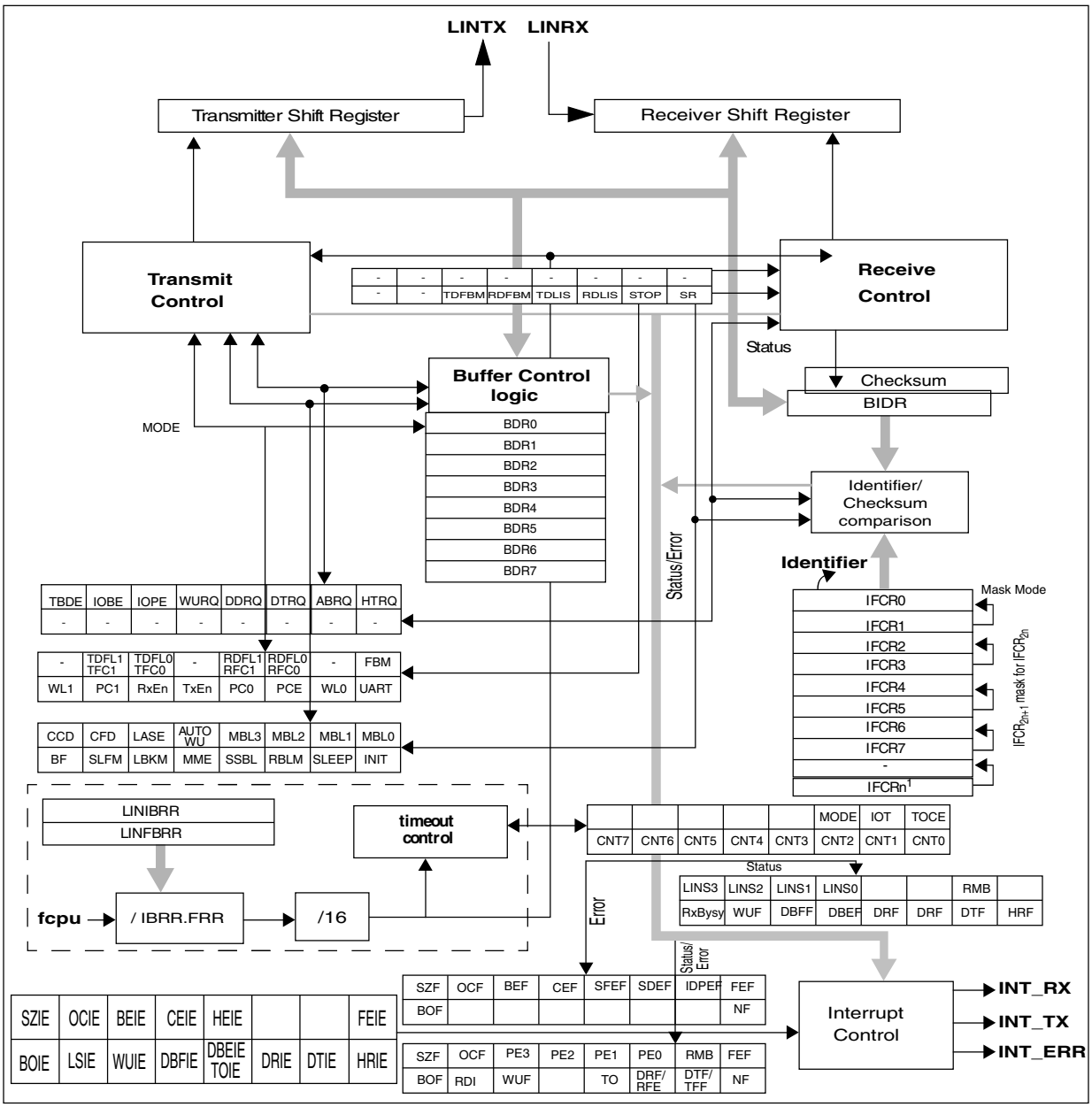


Figure 59-1. Block diagram

Notes:

1: The value of n depends on the no_of_filters. Refer to the chip configuration details to see the number of filters used in this device.

2: For LIN mode N = 16 and N is programmable by user in UART mode.

59.2 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features that are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test Mode:Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

59.2.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection

- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.

59.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers - see Baud rate generation
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate ≤ 6.25 Mbit/s
  - Sixteen times oversampling
  - 3:1 majority voting
- For 6.25 Mbit/s < bit rate ≤ 12.5 Mbit/s

- Reduced over sampling programmable by the user
- 3:1 majority voting for reduced over sampling of 8

- For 12.5 Mbit/s < bit rate ≤ 25 Mbit/s
  - Reduced over sampling programmable by the user
  - 1:1 voting for all reduced over sampling of 4, 5 and 6

59.3 Functional description

59.3.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

59.3.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

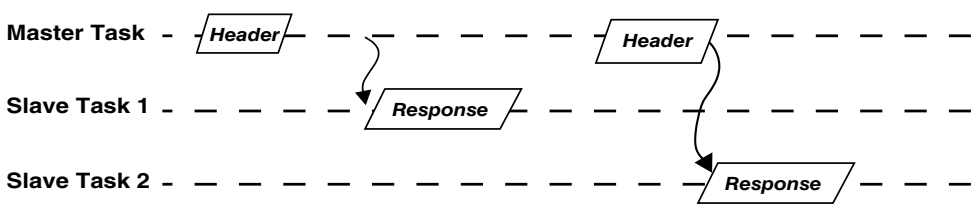


Figure 59-2. Frames

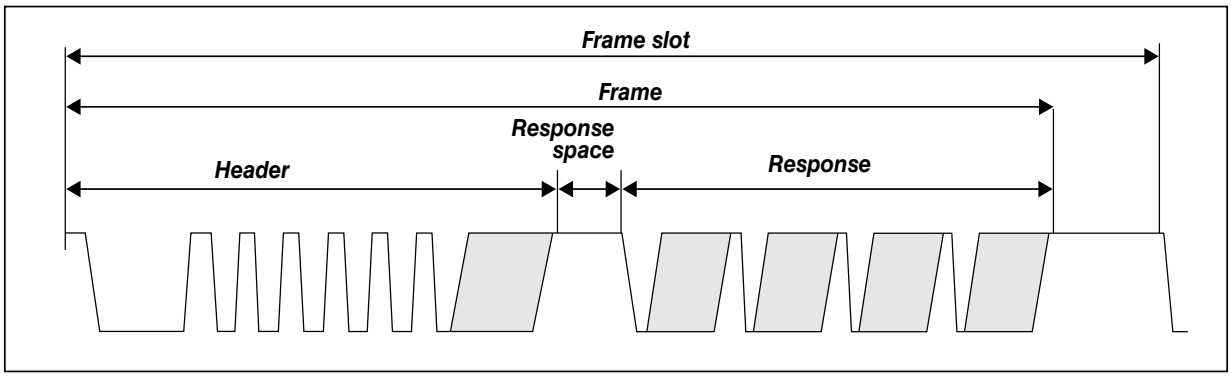


Figure 59-3. Structure of LIN frame

59.3.1.2 Data field

Each byte is transmitted as shown in Figure 59-4. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

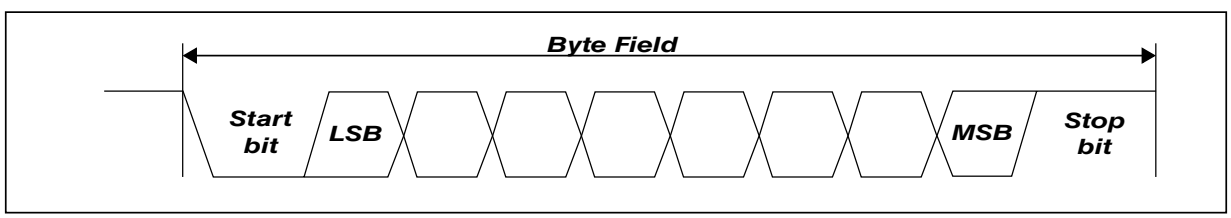


Figure 59-4. Structure of byte field

59.3.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in Figure 59-5. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

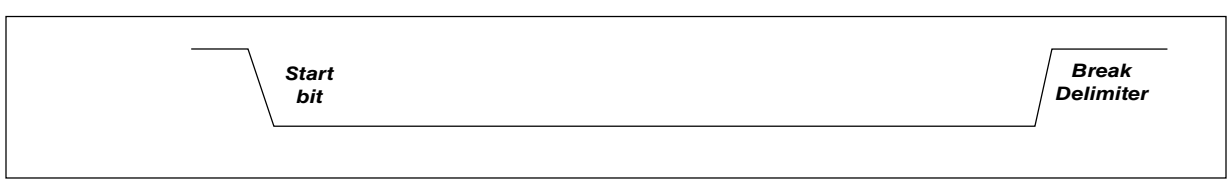


Figure 59-5. Break field

59.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

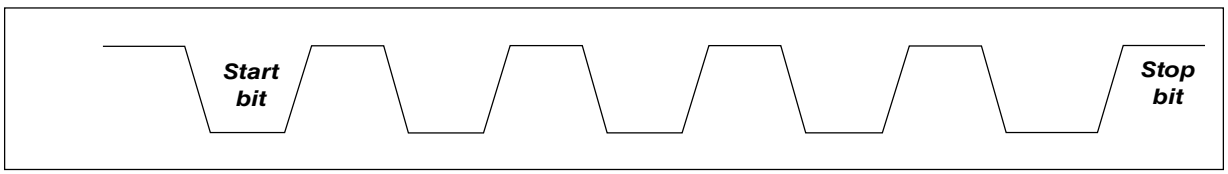


Figure 59-6. Sync byte field

59.3.1.5 Identifer

The Identifier field consists of two sub-fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

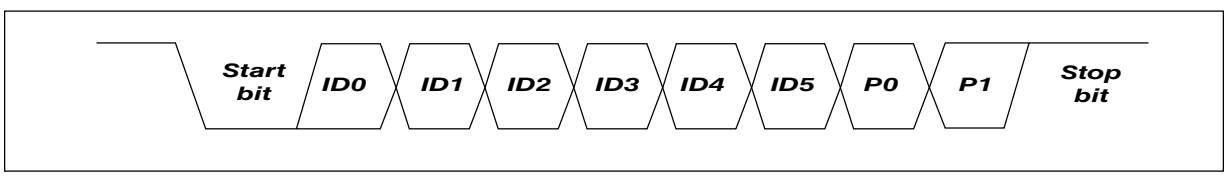


Figure 59-7. Identifier field

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

59.3.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and is used for communication with LIN 2.0 slaves.

59.3.2 LINFlexD features

59.3.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization, Normal** and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller software must:

1. Set up the baud rate registers
2. Reset UART bit
3. Select the mode (master or slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

Normal mode (NM)

Once software has completed initialization of the LINFlexD controller, it can enter the Normal mode by clearing the INIT bit.

Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be awakened from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to Wakeup management.

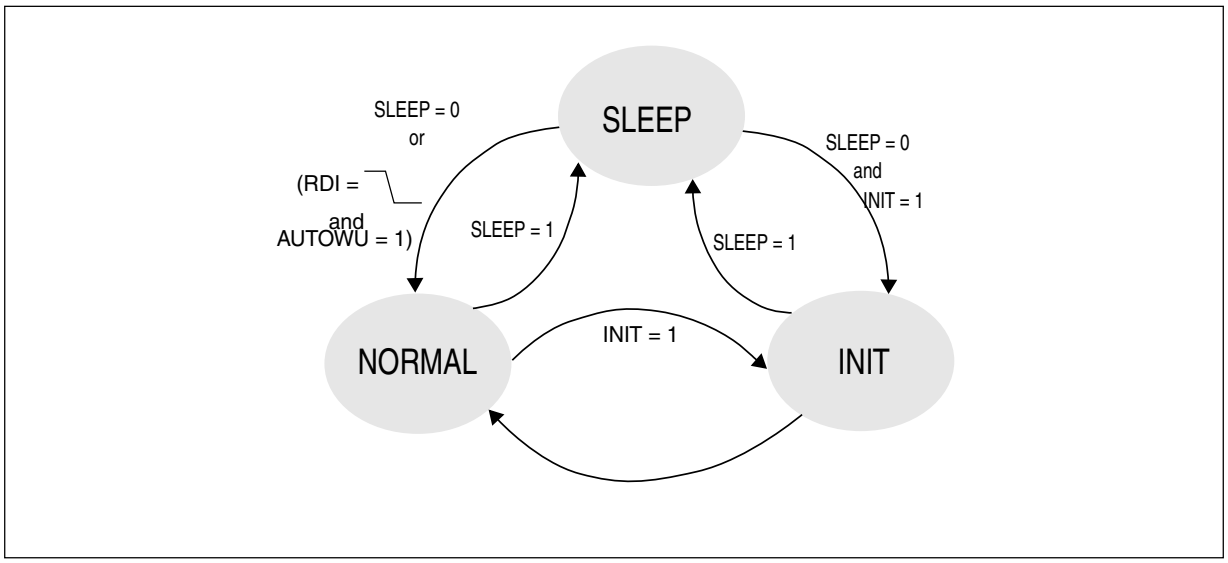


Figure 59-8. Operating modes

59.3.2.2 Test mode

Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is an internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

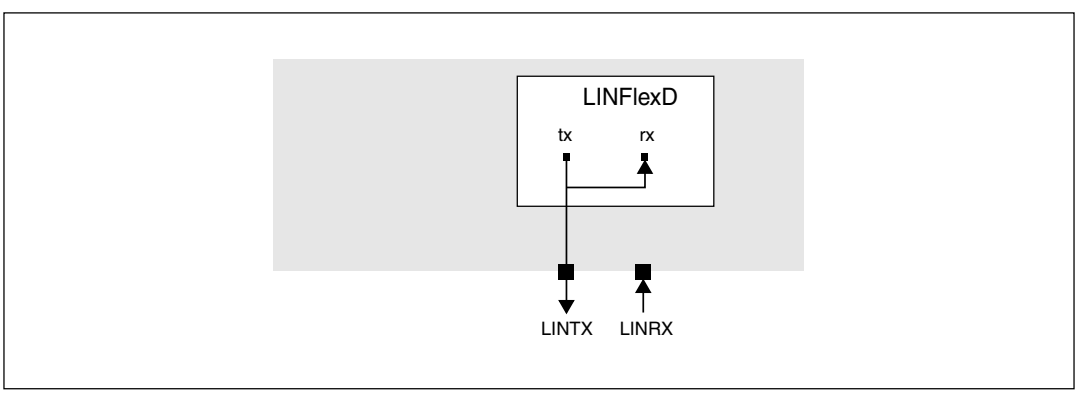


Figure 59-9. LINFlexD in Loop Back Mode

59.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits untill the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDRL and BDRM. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling).

It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to Error handling).

It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

59.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL[5:0] bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit, and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7].

The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software then you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

59.3.2.5 Errors

59.3.2.5.1 Header error

A header error is an error during the header reception. The error types are:

1. Sync Del error (SDEF)
2. Sync field error (SFEF)
3. Identifier Parity error (IDPEF)

Sync Del error (SDEF)

The delimiter should be 1 for at least one bit time; otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if *HEIE* bit of LINEIER is set.

Sync field error (SFEF)

The SFEF error condition is monitored differently depending on whether autosynchronization (LASE) is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the SFEF flag indicates:

- The deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators or
- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator).

This check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

If SFEF field is asserted it means that:

D > 14.0625%

If there is no error, it means that:

D < 14.84375%

If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

Overflow during Sync Field Measurement

This check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = '0')

In this case the Sync character is received as a normal character. If the received character is 0x55 then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

Identifier Parity error(IDPEF)

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

59.3.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CPU cycles. Bit error is not checked during break field transmission.

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) - (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note

If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

59.3.2.5.3 Framing error

This error is flagged when a dominant state is sampled on stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if FEIE bit is set in LINEIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

59.3.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINEIER.

59.3.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of slave, if buffer is not released (RMB is not reset) before reception of next Identifier and if RBLM is set, then the ID along with the data is discarded.

59.3.2.5.6 Timeout error

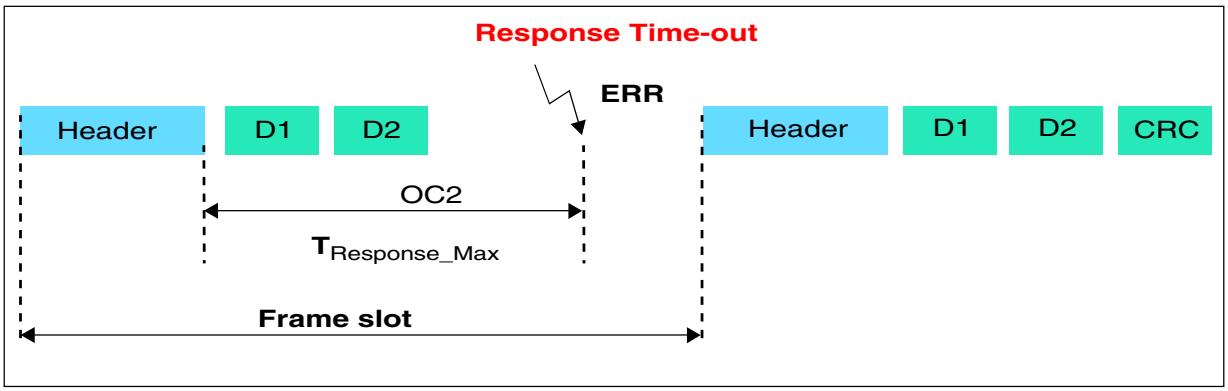


Figure 59-10. Incomplete response (for example, missing checksum)

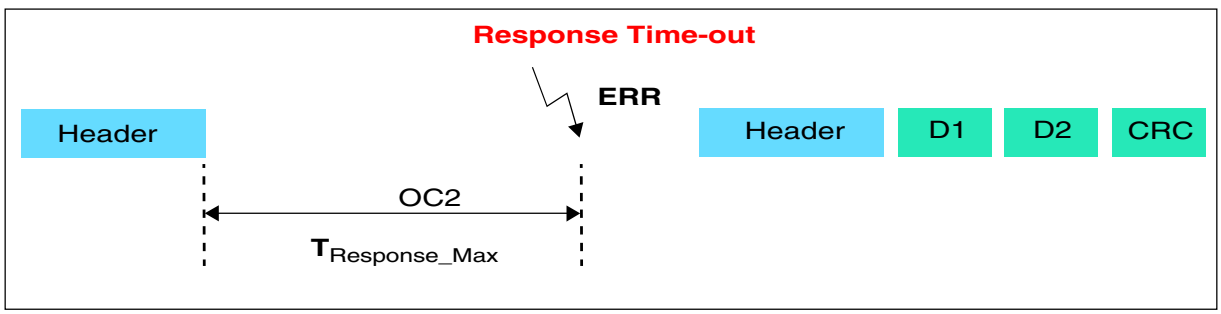


Figure 59-11. No response

Response timeout mechanism

- Master mode
  - OC2 is loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time).

    This loading takes place at the end of id field (and not at the beginning of data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further OC2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the OC2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the OC2 value (no change compared to earlier implementation).
- Slave mode

- **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, OC2 can be loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time))
- **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the implementation is the following:
  - At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT].
  - At the end of the first_data_byte it is reloaded again according to DFL
  - Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - no error on the bus and timing is correct (nominal header length),
  - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

  Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - header_max = 1.4 × Header_nominal = 49 Tbit
  - taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit
  - If the counter starts after 11 Tbit, the HTO value is 56–11 = 45 Tbit

  The reset value of HTO is 45, and this register can only be programmed in slave mode.

  As response space is not included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be reset and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

59.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8^{th}, 9^{th} and 10^{th} samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (see LIN Error Status Register (LINFlexD_LINESR).

59.3.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2n and 2n+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/ reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2n or IFCR2n+1 (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2n is the filter and 2n+1 acts as a mask for it. The FACT bit for filter 2n+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

If there is a match of identifier with the mth filter in any mode, then m+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only untill the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note

If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In mask mode if two filters match with the identifier then the filter having the lower number prevails.

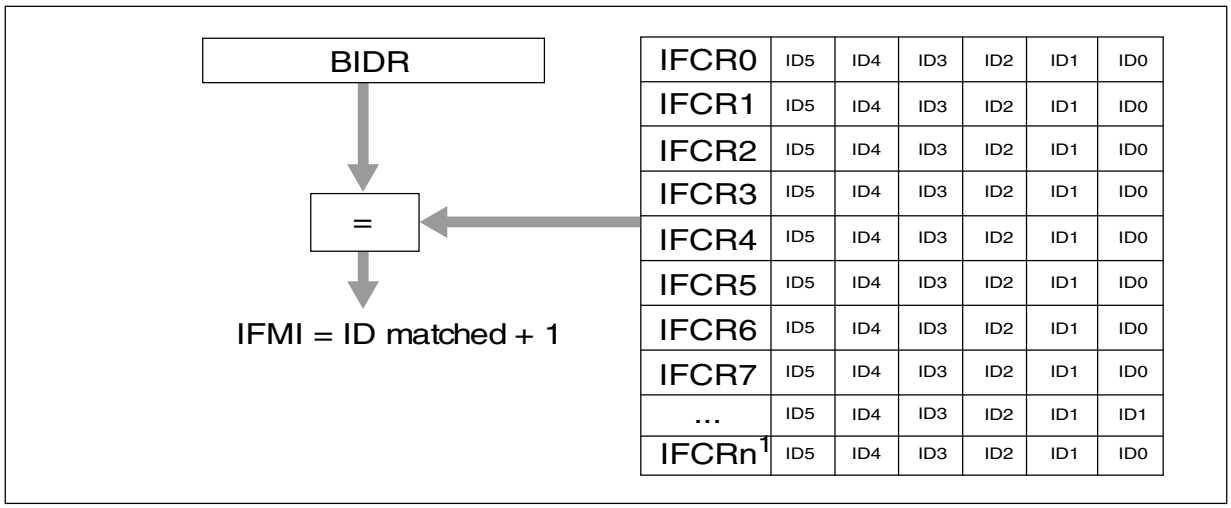


Figure 59-12. Identifier List mode

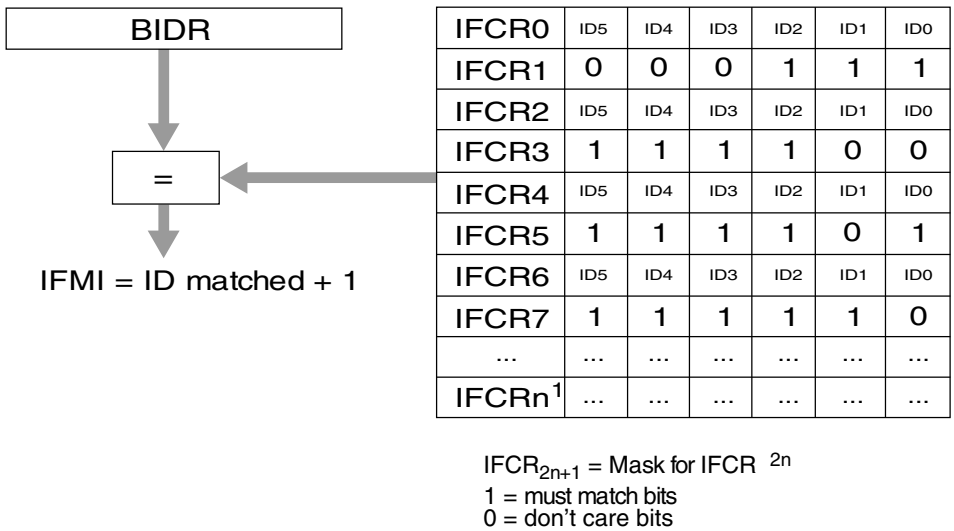


Figure 59-13. Identifier Mask mode

59.3.2.7 Start detection and break delimiter detection in receiver

There is a 10-bit-shift register sample register in the receiver that shifts signal data to the next least significant bit on each incoming sample.

Table 59-2. Start Detection and Delimiter Detection in receiver

9	8	7	6	5	4	3	2	1	0	Sample_reg
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3

Table continues on the next page...

Table 59-2. Start Detection and Delimiter Detection in receiver (continued)

9	8	7	6	5	4	3	2	1	0	Sample_reg
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x	0	x	0	x	0	start detect
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	delimiter detect
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

59.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to Table 59-2 for the status of the sample register (shift register) when count = 6

2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘0’, then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to ‘0’ then Noise flag is set.

59.3.2.7.2 Break delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to Table 59-2 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘1’, then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

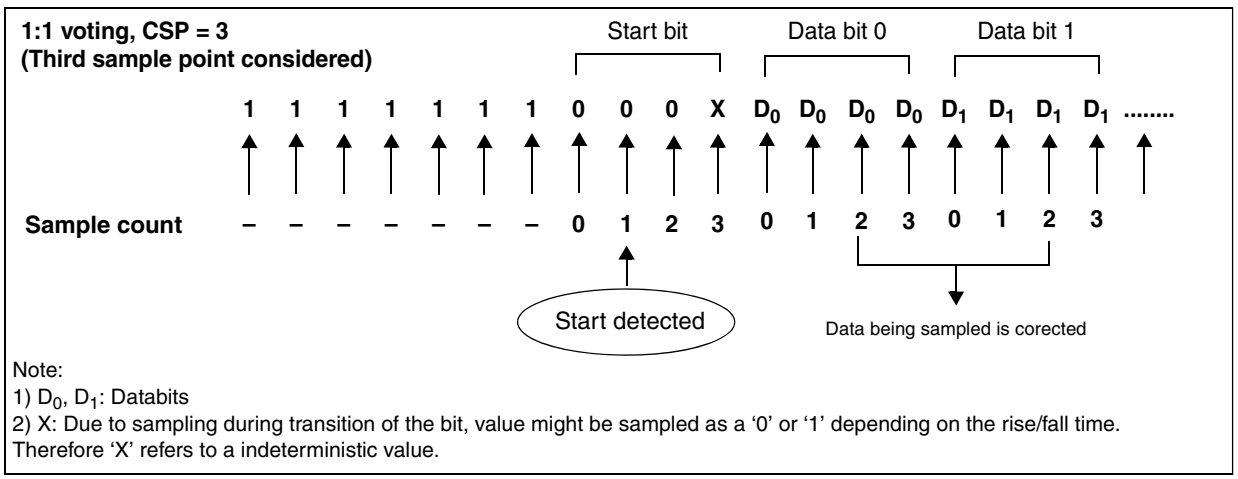


Figure 59-14. Start detection and sampling for over sampling rate = 4 (Case 1)

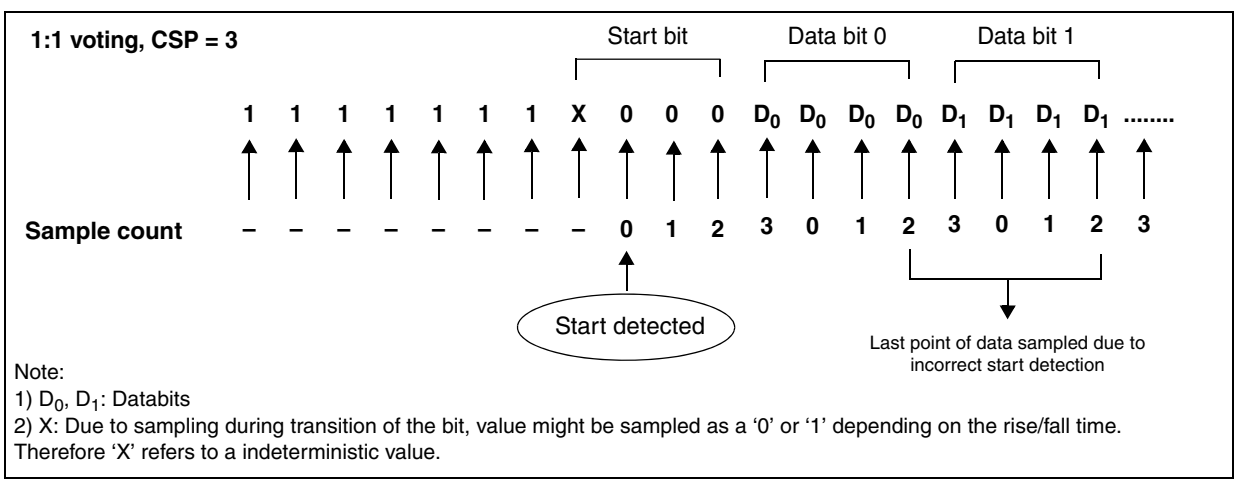


Figure 59-15. Start detection and sampling for over sampling rate = 4 (Case 2a)

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

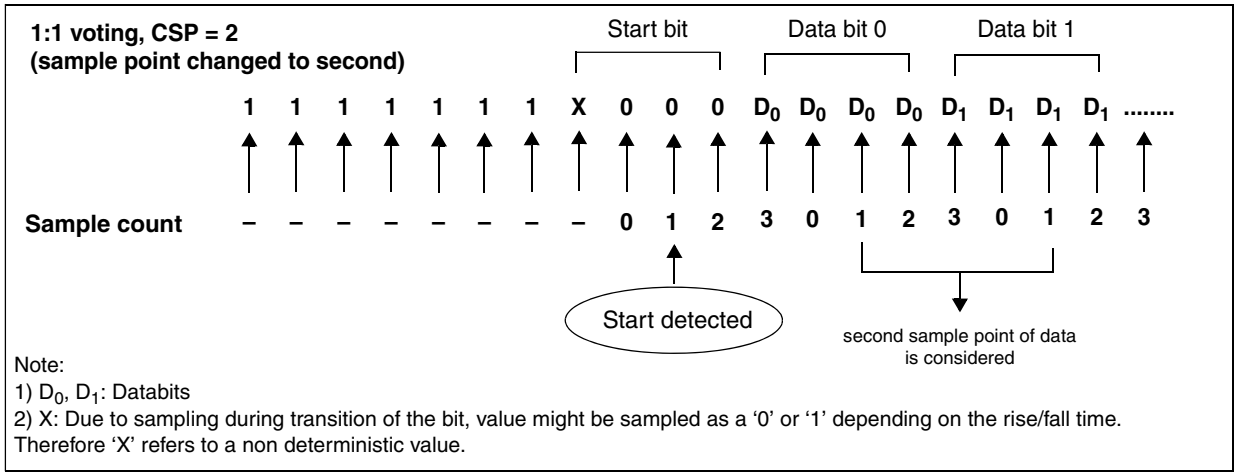


Figure 59-16. Start detection and sampling for over sampling rate = 4 (Case 2b)

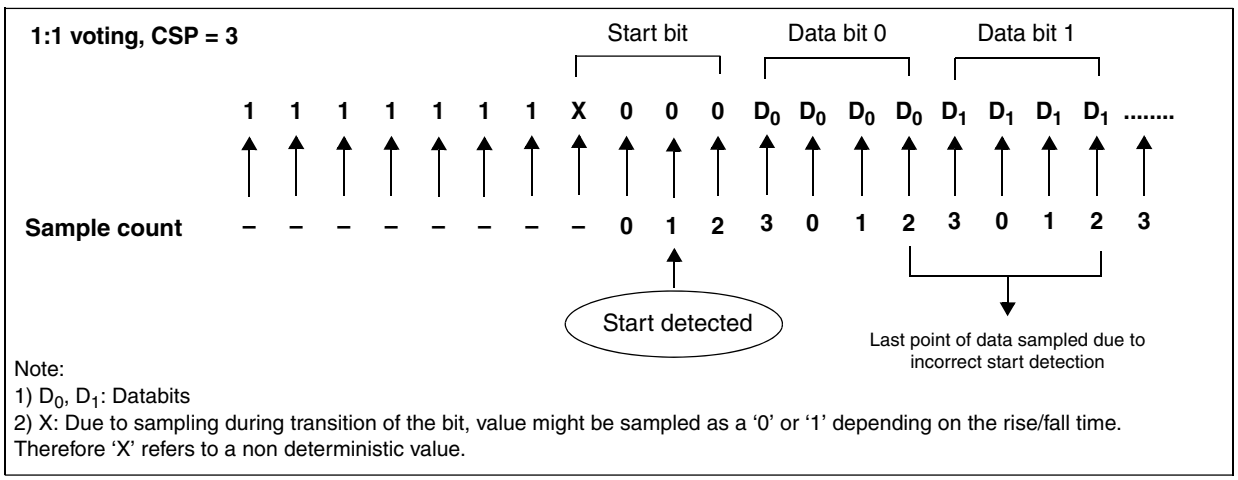


Figure 59-17. Start detection and sampling for over sampling rate = 4 (Case 3)

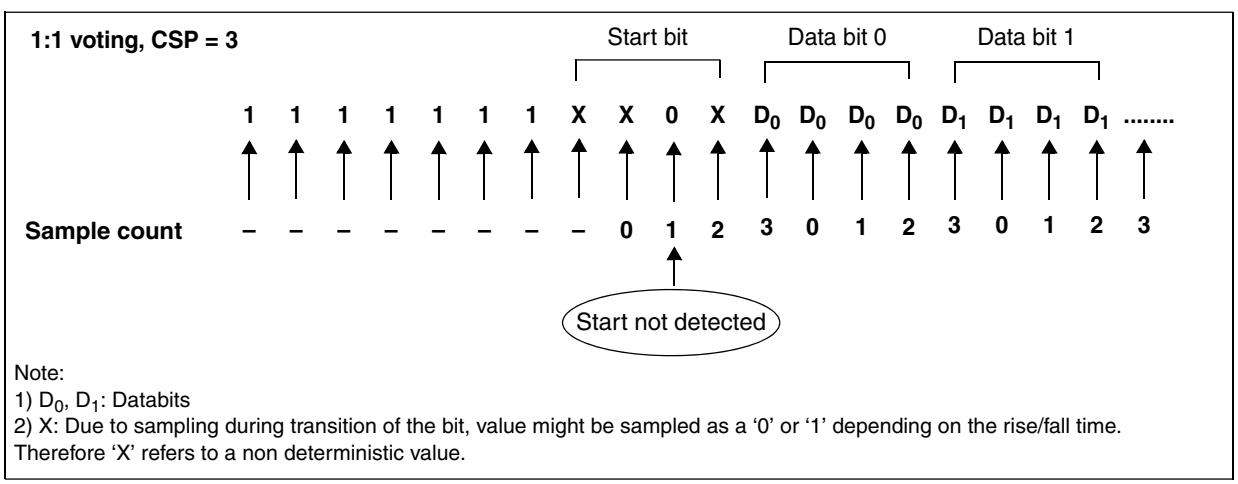


Figure 59-18. Start detection and sampling for over sampling rate = 4 (Case 4)

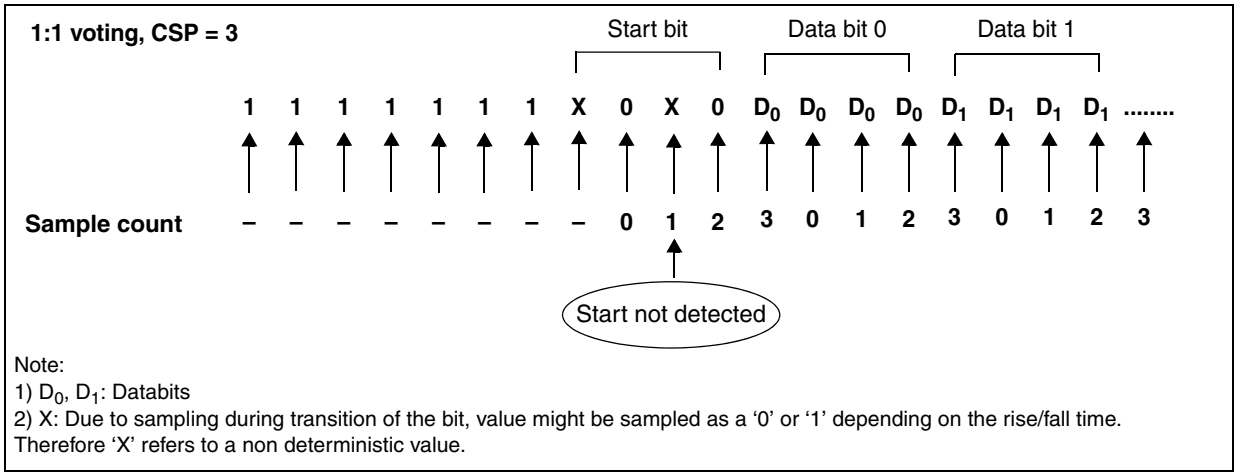


Figure 59-19. Start detection and sampling for over sampling rate = 4 (Case 5)

59.3.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate Register and the LIN Fraction baud rate register. The Baud Rate Registers can be programmed only during Initialization mode.

Baud rate is calculated with the following formula for both receiver and transmitter.

When ROSE = 1, Tx = Rx = LIN_CLK / (OSR × LDIV). When ROSE = 0, Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock.

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled, LINFBRR should not be used and programmed to zero and LDIV contains only the integer part of LINIBRR.

For example: When ROSE = 0 (For LIN and UART mode): LDIV = 468.75 d, LIN_CLK = 36 MHz, LINIBRR = 468 d, LINFBRR = 12 *Baud rate = 36 MHz/ (16 × 468.75) = 4.8 Kbit/s*

For example: When ROSE = 1 (Only for UART mode): LDIV = 10 d, LIN_CLK = 80 MHz, LINIBRR = 4 d, OSR = 4, *Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s*

59.3.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescalar value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When auto synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

59.3.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 μs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 μs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

1. AUTOWU = 1 On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated if WUPIE is set. LINFlexD is now in normal mode and ready to receive frames.
2. AUTOWU = 0 On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE bit is set). It is then up to the software to clear the SLEEP bit.

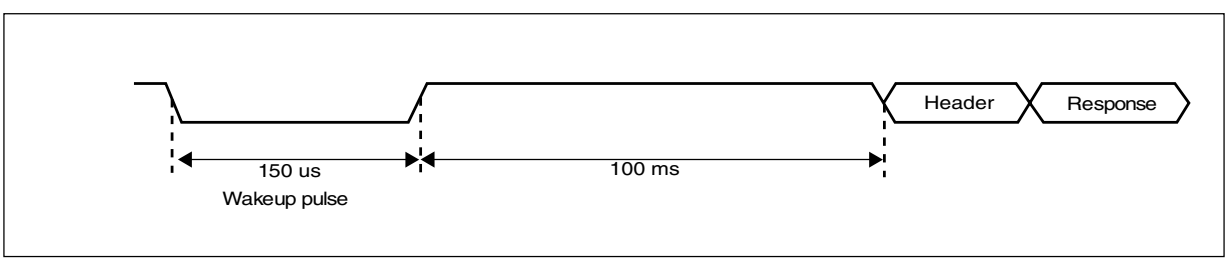


Figure 59-20. Wakeup sequence

59.3.3 Timer

There is an 8-bit counter that is free-running (even in Sleep and Initialization modes) if it is configured in Output Compare mode. There are two software configurable-output compare registers that, upon match with counter value, generate output compare interrupt provided TOCE bit in LINTCSR is set.

If this timer is configured in LIN mode, then software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

59.3.4 UART mode

Main features in the UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames, 13-bit frames, 16-bit frames, 17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

59.3.4.1 8-bit data frames

The eighth bit can be a data or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

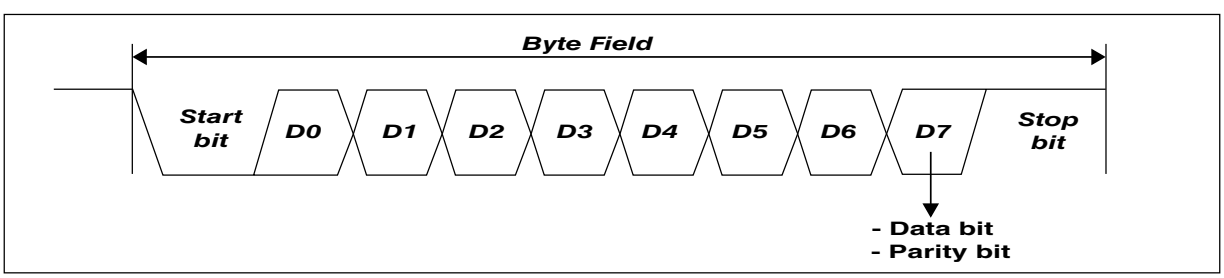


Figure 59-21. UART mode 8-bit data frame

59.3.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

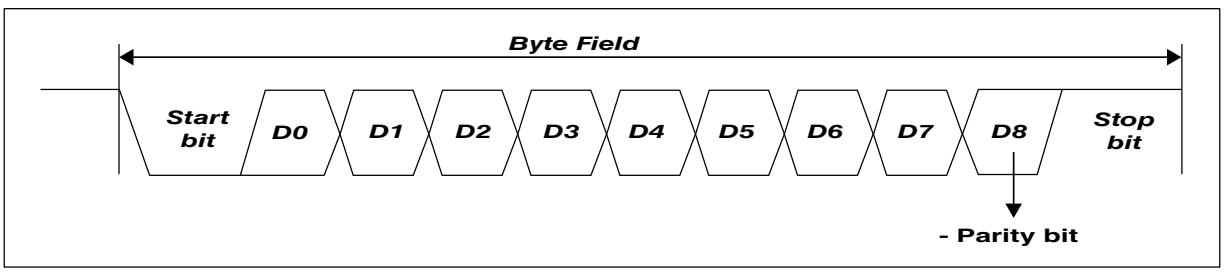


Figure 59-22. UART mode 9-bit data frame

59.3.4.3 16-bit data frames

The sixteenth bit can be a data or a parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

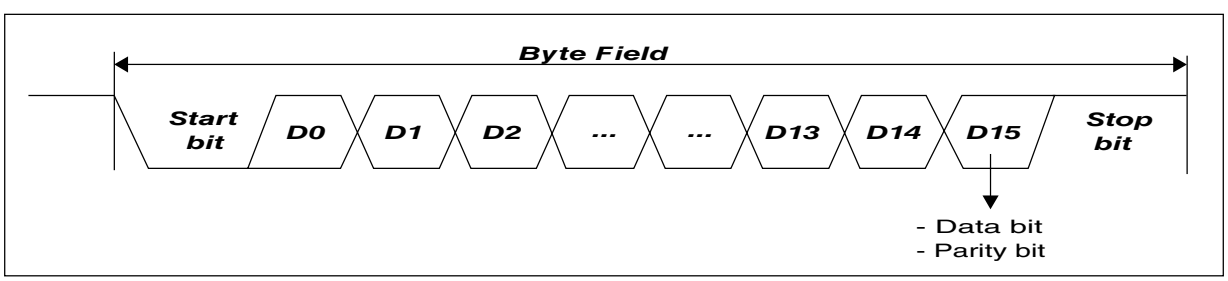


Figure 59-23. UART mode 16-bit data frame

59.3.4.4 17-bit frames

The seventeenth bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

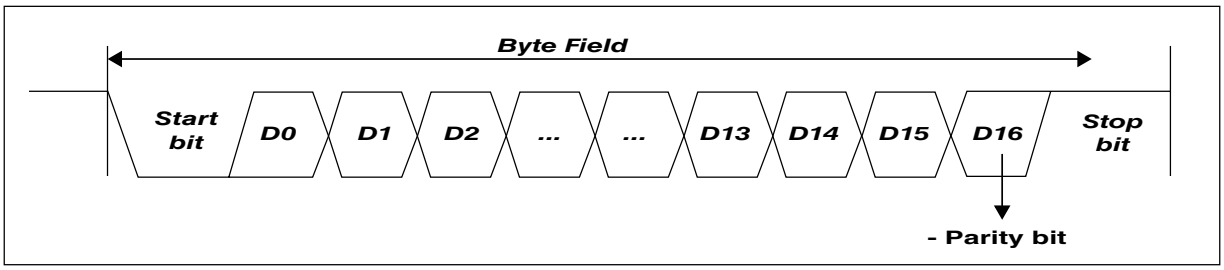


Figure 59-24. UART mode 17-bit data frame

59.3.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

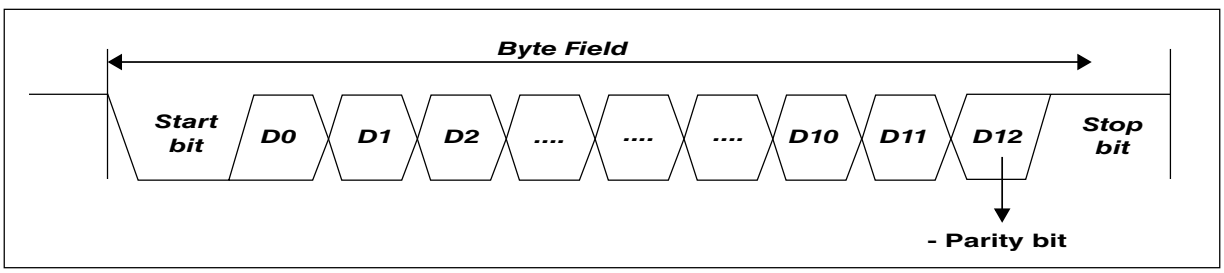


Figure 59-25. UART mode 13-bit data frame

59.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in the following figure:

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

Figure 59-26. Structure of 8-byte buffer

For 16-bit frames, the lower eight bits are written in BDR0 and upper eight bits in BDR1.

The same applies for reception.

59.3.4.7 UART transmitter

UART transmitter

In order to start transmission in the UART mode, UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the TxEn bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

NOTE

If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Table 59-3. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Byte0	BDRL	FIFO	Byte	OK
Write Byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write Half-word0-1	BDRL	FIFO	Byte	IPS transfer error
Write Word	BDRL	FIFO	Byte	IPS transfer error
Write Byte0-1-2-3	BDRL	FIFO	Half-word	IPS transfer error
Write Half-word0	BDRL	FIFO	Half-word	OK
Write Half-word1	BDRL	FIFO	Half-word	IPS transfer error
Write Word	BDRL	FIFO	Half-word	IPS transfer error
Read Byte0-1-2-3	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Half-word0-1	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Word	BDRL	FIFO	Byte/Half-word	IPS transfer error
Write Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Write Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Write Word	BDRL	BUFFER	Byte/Half-word	OK
Read Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Read Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Read Word	BDRL	BUFFER	Byte/Half-word	OK

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

59.3.4.8 UART receiver

UART receiver

The reception of a data byte is started as soon as the user exits initialization mode, sets the RxEn bit, and detects start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-half-words (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. RxEn bit needs to be set only to start the reception.The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 59-4. BDRM access in UART mode

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Byte4	BDRM	FIFO	Byte	OK
Read Byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read Half-word2-3	BDRM	FIFO	Byte	IPS transfer error
Read Word	BDRM	FIFO	Byte	IPS transfer error
Read Byte4-5-6-7	BDRM	FIFO	Half-word	IPS transfer error
Read Half-word2	BDRM	FIFO	Half-word	OK
Read Half-word3	BDRM	FIFO	Half-word	IPS transfer error
Read Word	BDRM	FIFO	Half-word	IPS transfer error
Write Byte4-5-6-7	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Word	BDRM	FIFO	Byte/Half-word	IPS transfer error
Read Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	OK
Read Half-word2-3	BDRM	BUFFER	Byte/Half-word	OK
Read Word	BDRM	BUFFER	Byte/Half-word	OK
Write Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Word	BDRM	BUFFER	Byte/Half-word	IPS transfer error

Note

Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and data bits of the registers BDRL/BDRM.

Note

- If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.
- If RDFL is programmed for a certain value but that number of bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to Idle state.
- If a STOP request arrives in the middle of one reception, it is only acknowledged after all the programmed number of data bytes have received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout. When the state machine moves to Idle state, then only a stop request is served.
- If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if FEIE bit in LINIER is set. As there is only one register bit for framing error, this interrupt will be helpful in identifying which byte has framing error.
- If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is depending on RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.
- When WLS bit = 1, BDRM access depends on WL bit setting.

- When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.
- When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

59.3.5 DMA interface

59.3.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

59.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in Table 59-5 and Table 59-6.

Table 59-5. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)

Table continues on the next page...

Table 59-5. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
BIDR	DFL [5:0]	—	Data field length: • 1 to 8 bytes: normal frames • 1 to 64 bytes: extended frames
	DIR	—	Direction of the data field: • 0 => reception • 1 => transmission
	CCS	—	Checksum type: • 0 enhanced • 1 classic
	ID[5:0]	—	Identifier.
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA. • 0 buffer data is free • 1 buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7. Used for the extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7. Used for the extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received. Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode).
UARTCR	FBM	—	FIFO/Buffer mode • 0 Buffer mode • 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty • 0 Rx FIFO not empty • 1 Rx FIFO empty
	TFF	—	Tx FIFO full • 0 Tx FIFO not full • 1 Tx FIFO full

DMATXE[2**TX_CH_NUM][1]		High	DMA Tx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA TX channel = IFMI[3:0] -1.
			DMATXE[x] = 0b => TX channel x disabled
			DMATXE[x] = 1b => TX channel x enabled

DMARXE[2**RX_CH_NUM][1]		High	DMA Rx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.

Table 59-5. LINFlexD control/status fields description

Register	Field	Level	Description

			In LIN slave mode: # DMA RX channel = IFMI[3:0] -1.
			DMARXE[x] = 0b => RX channel x disabled
			DMARXE[x] = 1b => RX channel x enabled

1. ** stands for exponentiation.
1. ** stands for exponentiation

Table 59-6. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0]	High	Logical AND between the 2 DMA enable bits (LINFlexD and eDMA).
DMA_TEN	LIN slave Tx	DMATXE[x] x = IFMI — 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] x = IFMI — 1	High	

Control/status fields of the TCD descriptors referred to in this document are described in Table 59-7.

Table 59-7. TCD control fields description

TCD Field	Level	Description
CITER[14:0]	—	Current "major" iteration count
BITER[14:0]	—	Beginning "major" iteration count
NBYTES[31:0]	—	Inner "minor" byte transfer count. Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address

Table continues on the next page...

Table 59-7. TCD control fields description (continued)

TCD Field	Level	Description
DOFF[15:0]	—	Destination address signed offset applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request.
DONE	High	Channel done (the DMA has completed the outer major loop).
D_REQ	High	Disable request. If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

59.3.5.3 Master node –TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

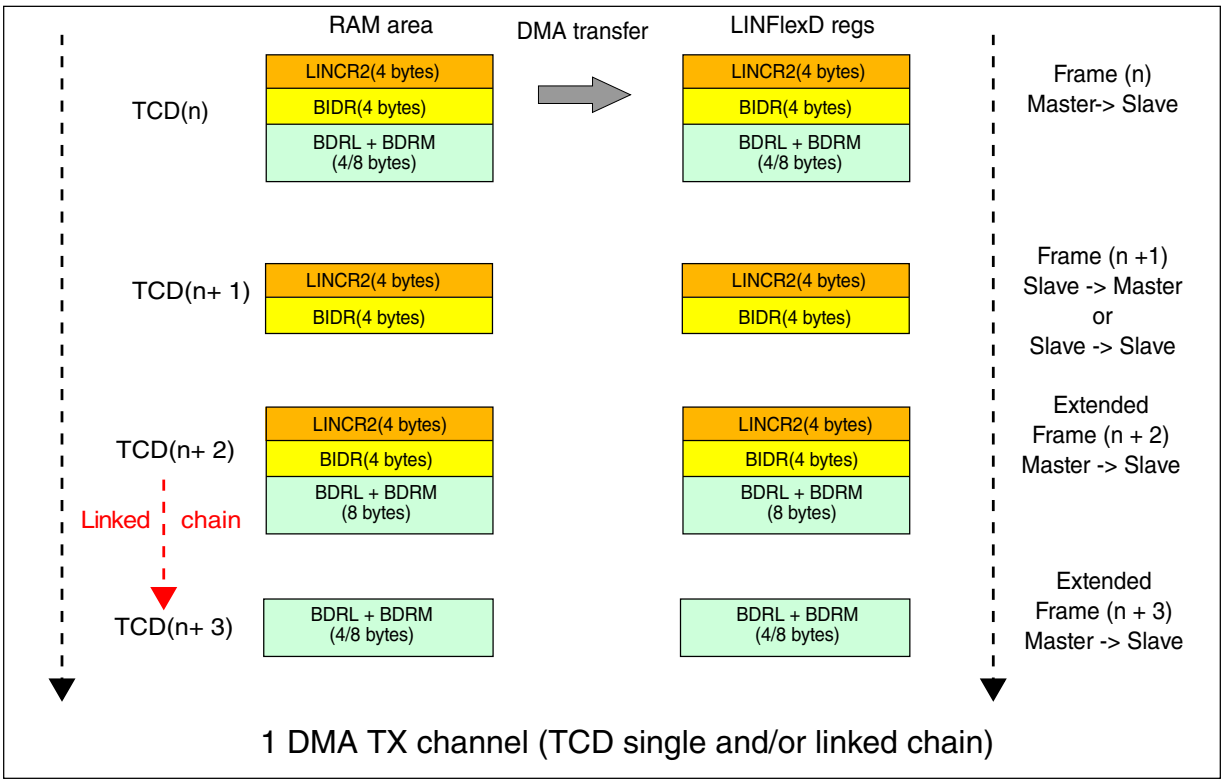


Figure 59-27. Master node –TX memory map

The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

The following table provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 59-8. Master node – Tx mode – Register setting

LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

Table continues on the next page...

Table 59-8. Master node – Tx mode – Register setting (continued)

LIN frame	LINCR2	BIDR
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in Table 59-9. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 59-9. TCD setting – Master node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	-	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	
DADDR[31:0]	-	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

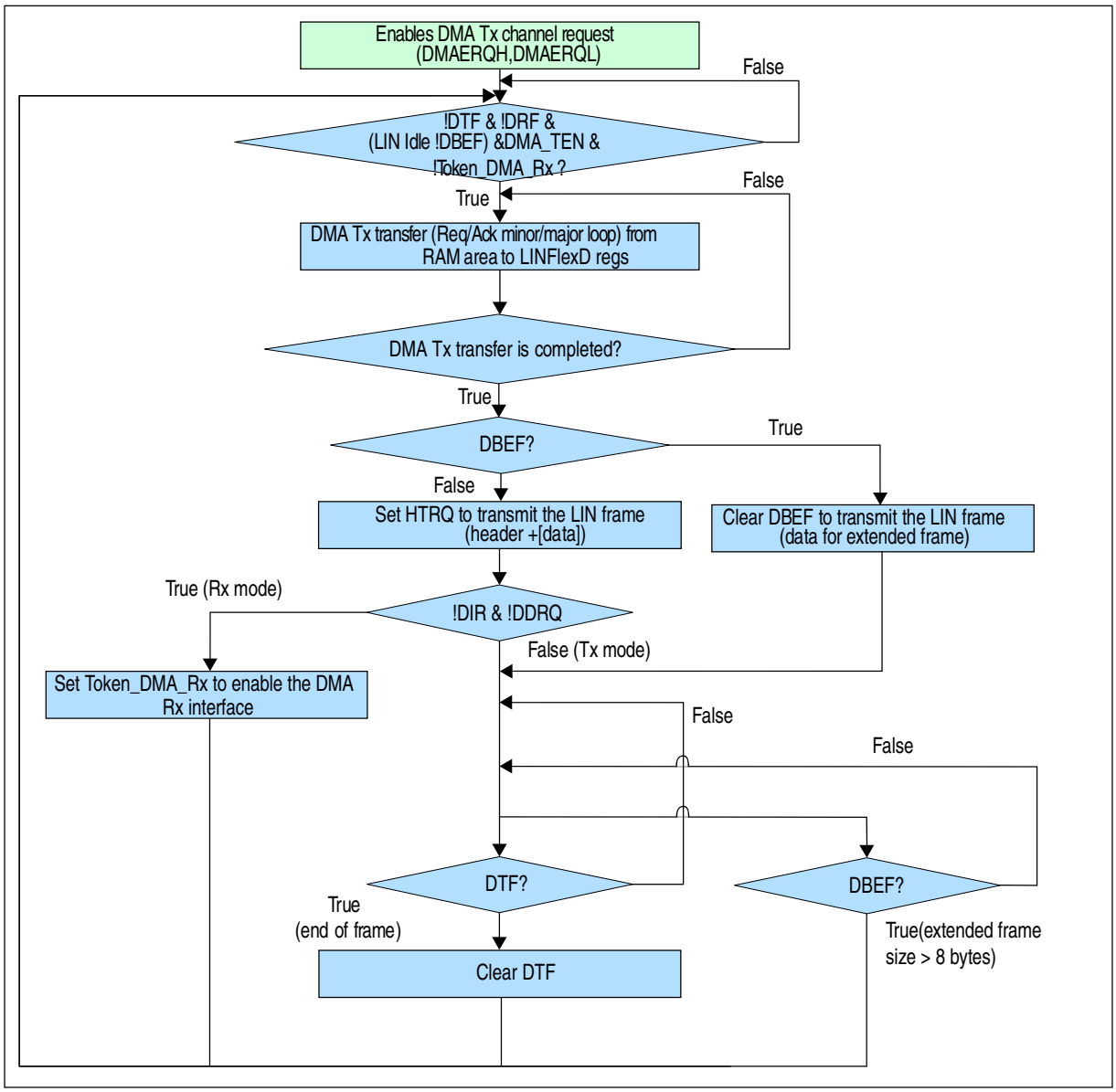


Figure 59-28. Master node – DMA Tx FSM (concept scheme)

59.3.5.4 Master node - RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame , except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

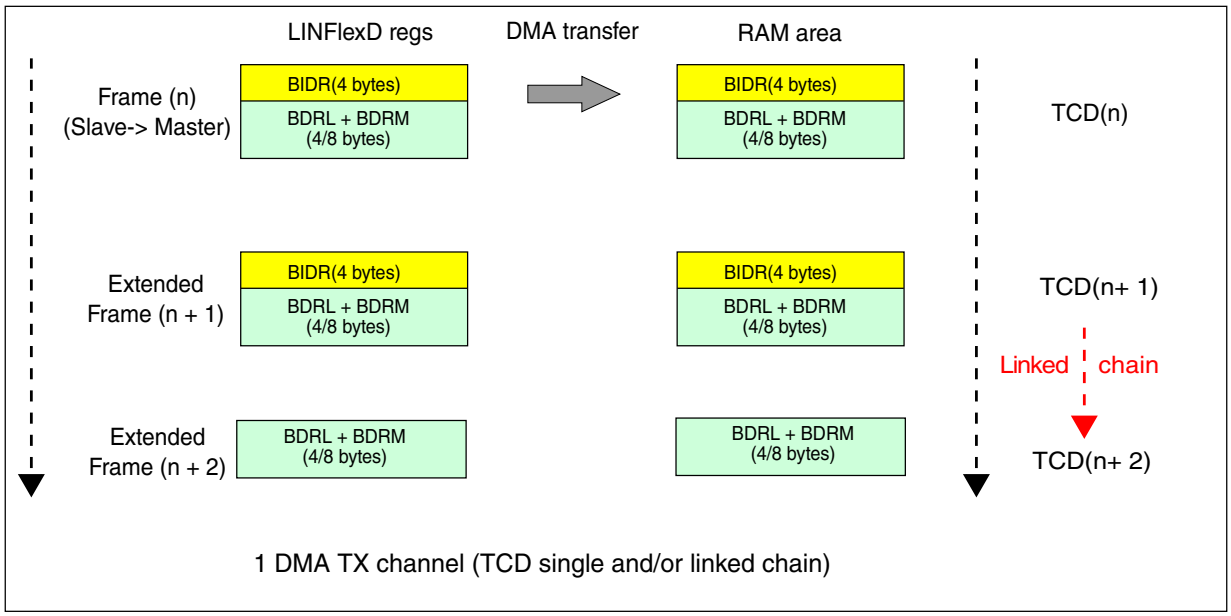


Figure 59-29. Master node – RX memory map

The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in the following figure. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in the next table. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 59-10. TCD setting – Master node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer

Table continues on the next page...

Table 59-10. TCD setting – Master node – Rx mode (continued)

TCD Field	Value	Description
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

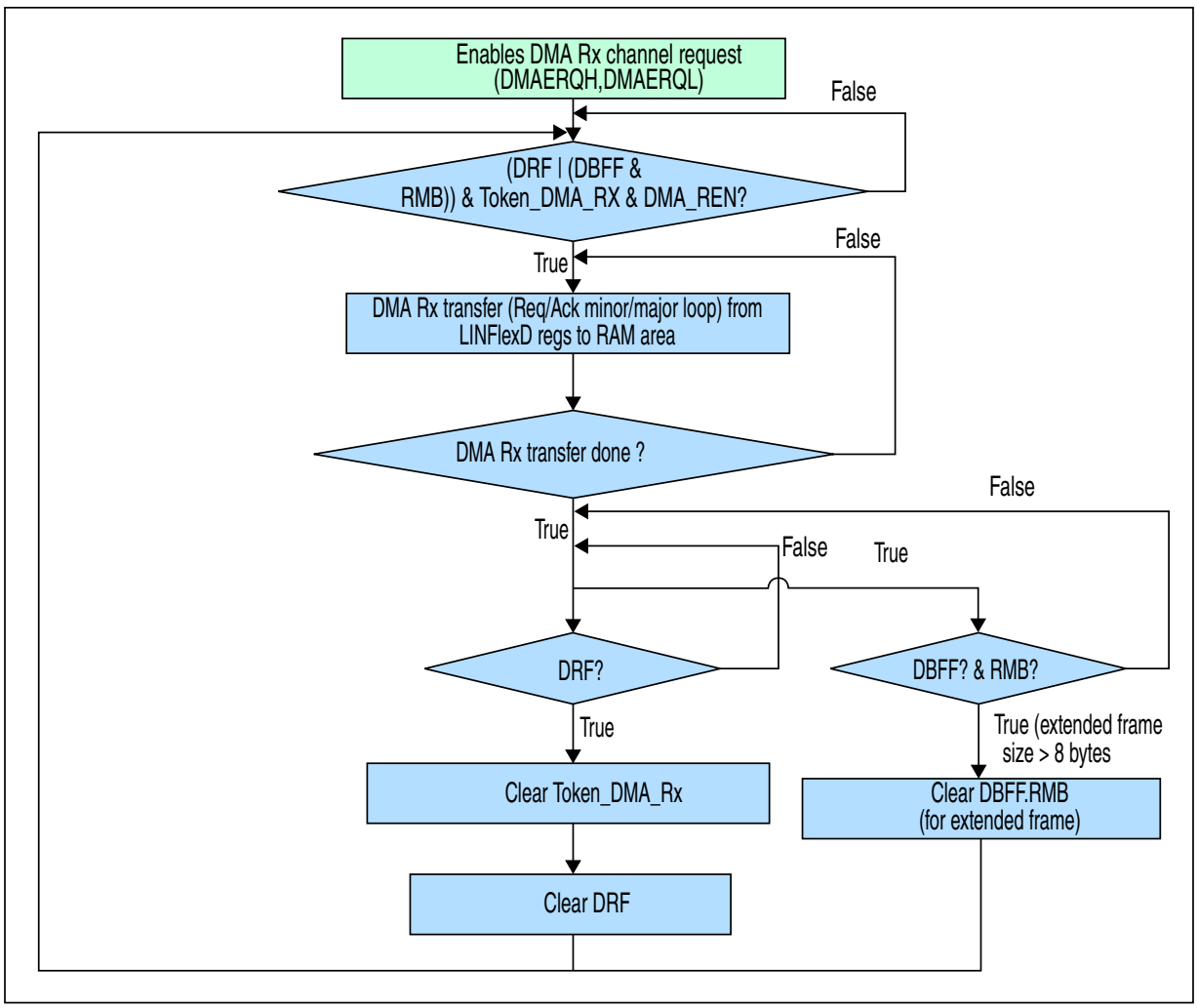


Figure 59-30. Master node – DMA Rx FSM (concept scheme)

59.3.5.5 Slave node – TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

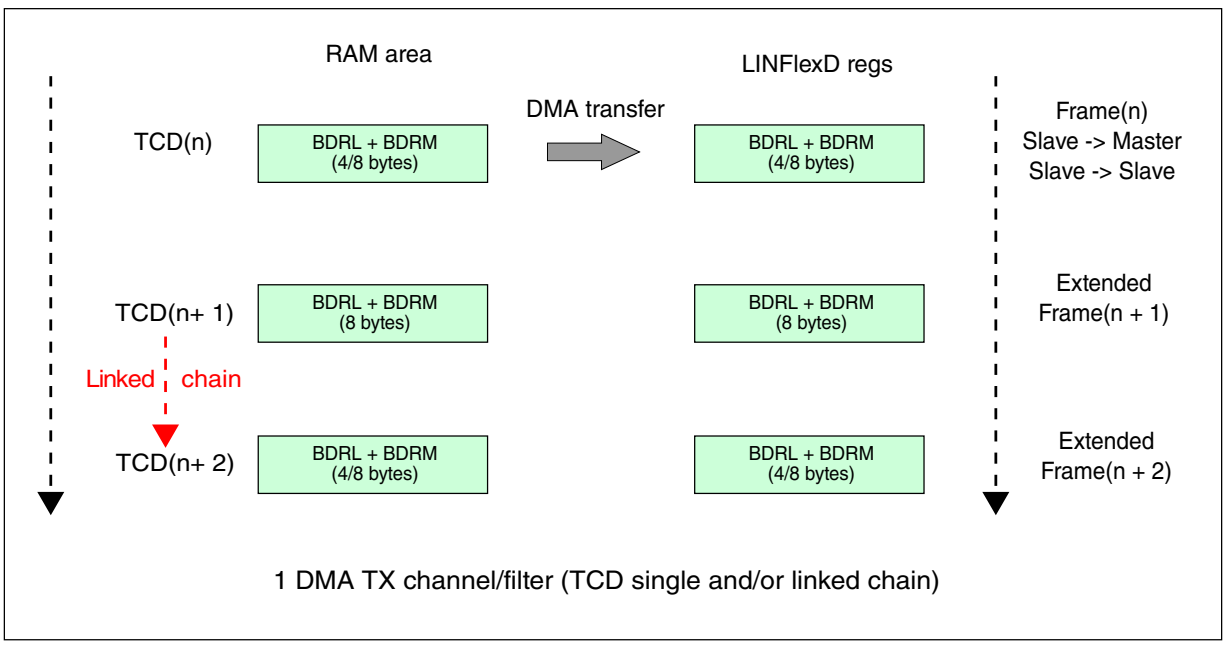


Figure 59-31. Slave node – TX memory map

The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in Table 59-11.

Table 59-11. Slave node – Tx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1.The TCD setting (word transfer) is given in Table 59-12. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 59-12. TCD setting – Slave node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

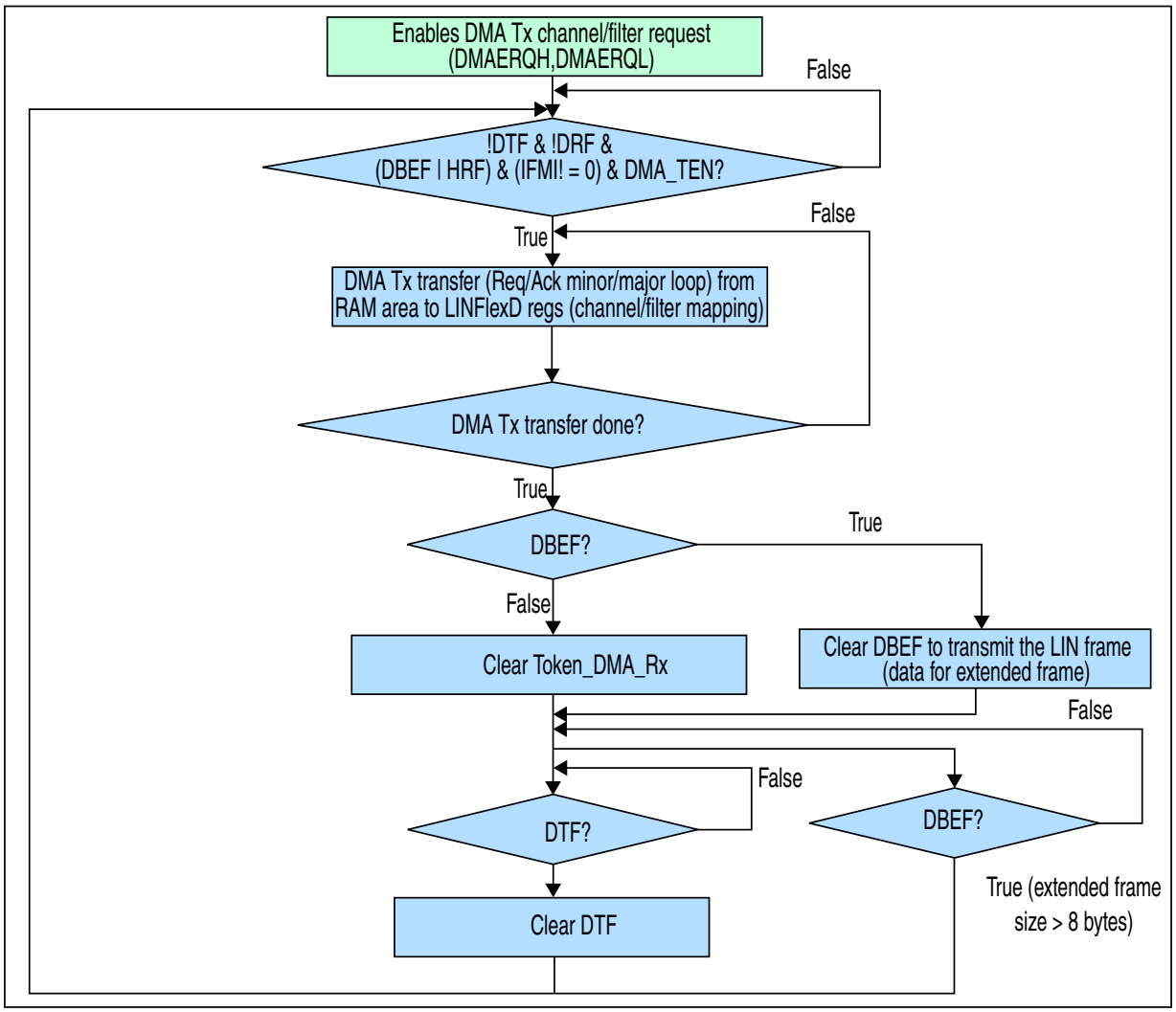


Figure 59-32. Slave node – DMA Tx FSM (concept scheme)

59.3.5.6 Slave node – RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

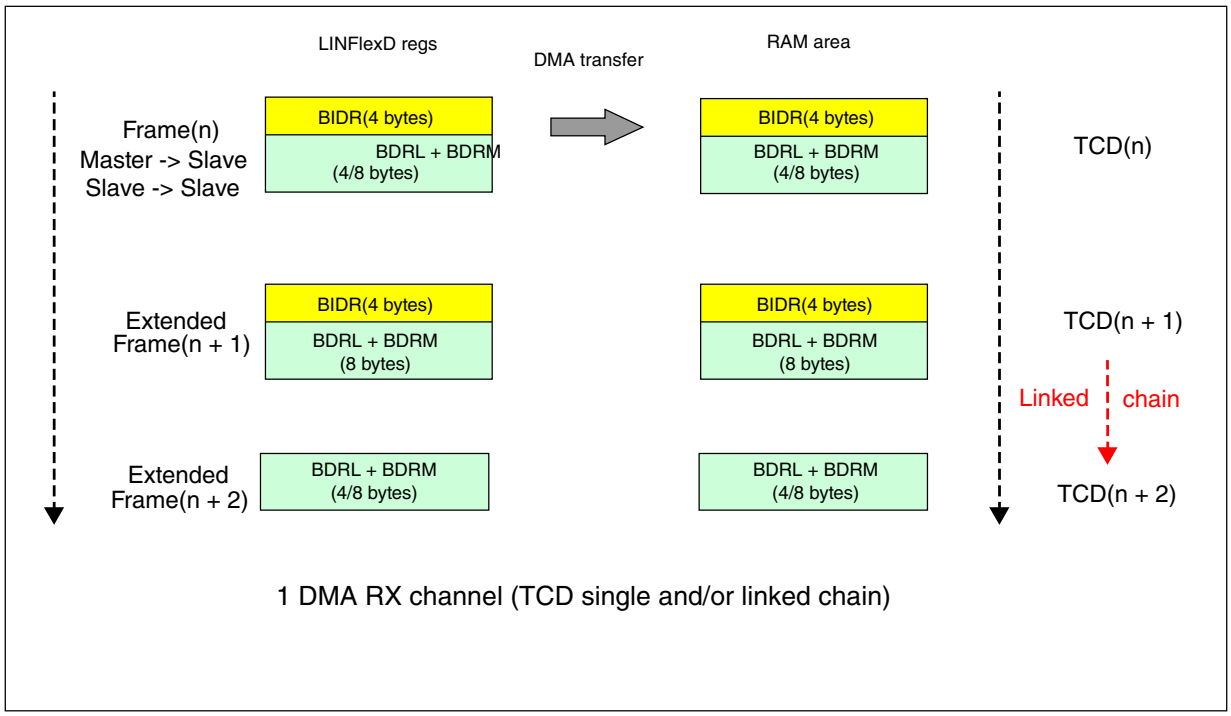


Figure 59-33. Slave node – RX memory map

The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, IFCR registers is given in Table 59-13.

Table 59-13. Slave node – Rx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in the following figure. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in Table 59-14. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 59-14. TCD setting – Slave node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

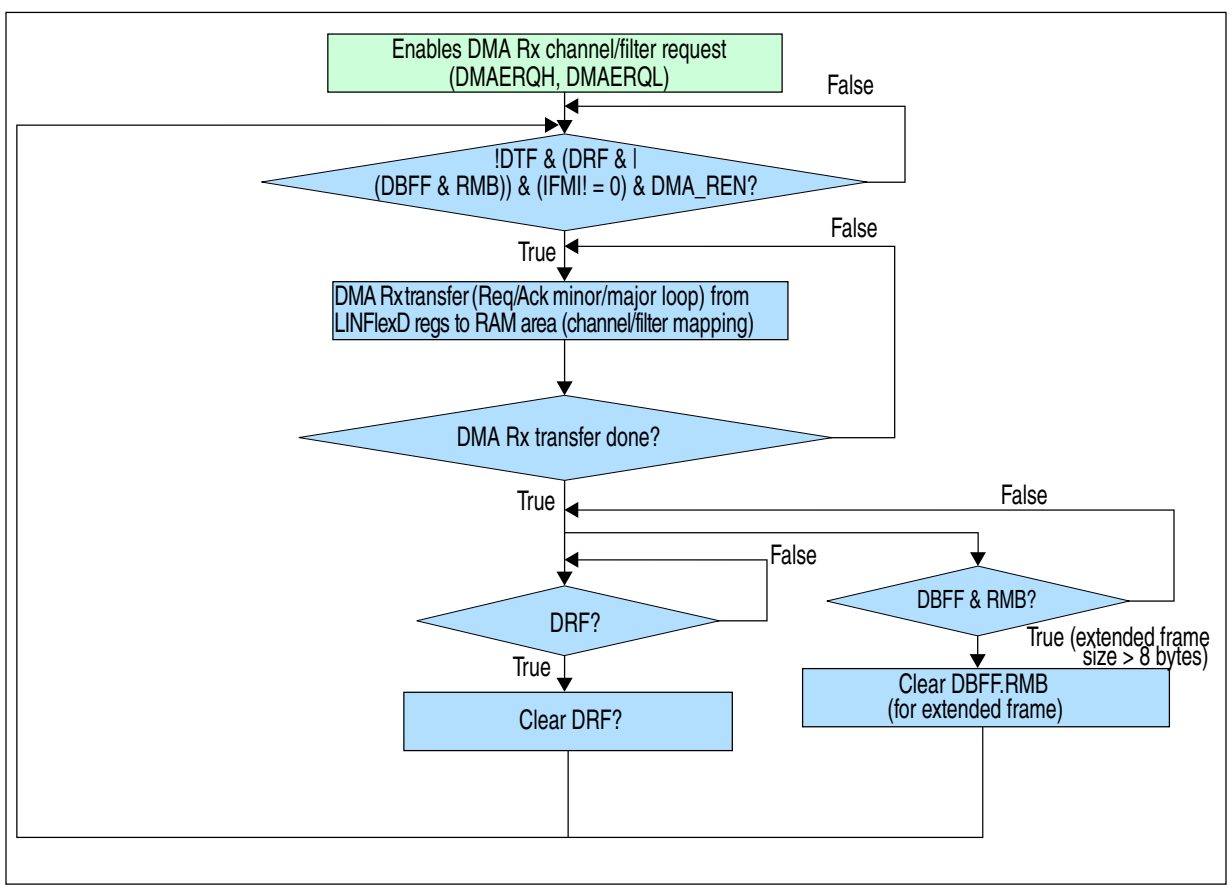


Figure 59-34. Slave node – DMA Rx FSM (concept scheme)

59.3.5.7 UART – TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Figure 59-35.

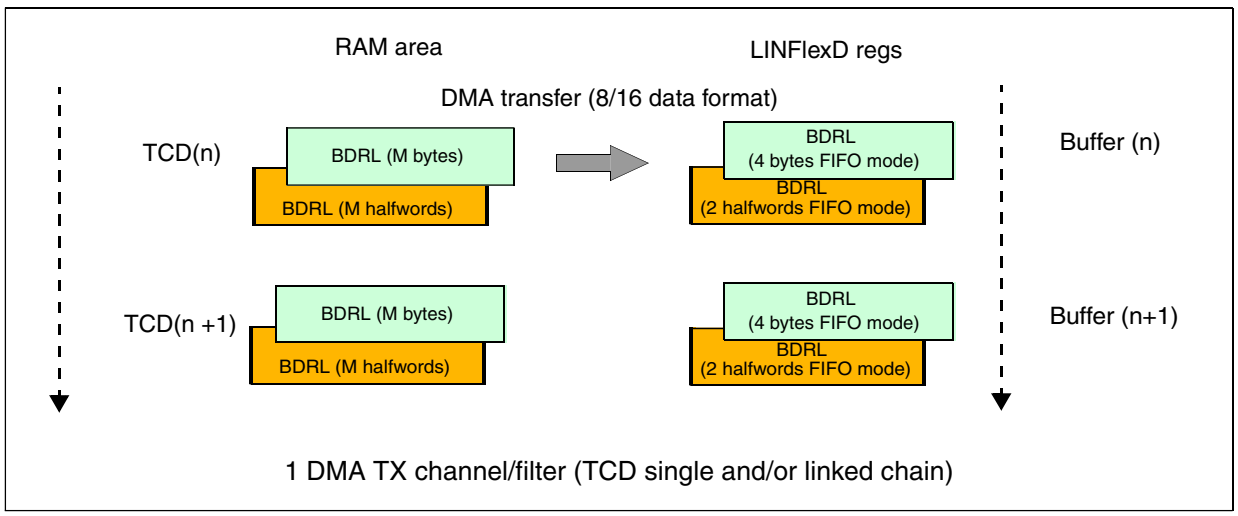


Figure 59-35. UART – TX memory map

The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8 bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

The concept FSM to control the DMA Tx interface is given in Figure 59-36. DMA Tx FSM will move to Idle state if DMATXE[0] = 0.The TCD setting (typical case) is given in Table 59-15. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 59-15. TCD setting – UART – Tx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		RAM address
SOFF[15:0]	1	2	Byte/Half-word increment
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	–M	–M × 2	-
DADDR[31:0]	-		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

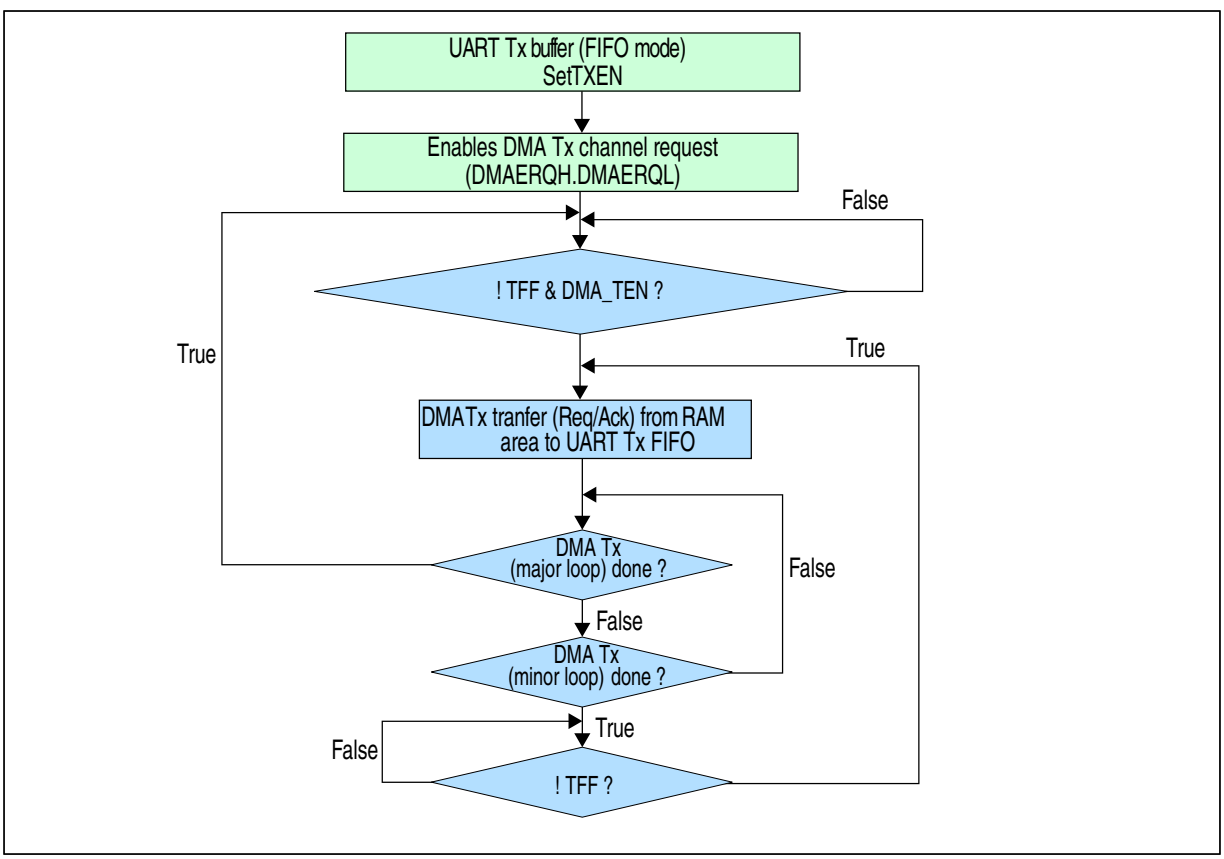


Figure 59-36. UART – DMA Tx FSM (concept scheme)

59.3.5.8 UART – RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Table 59-16.

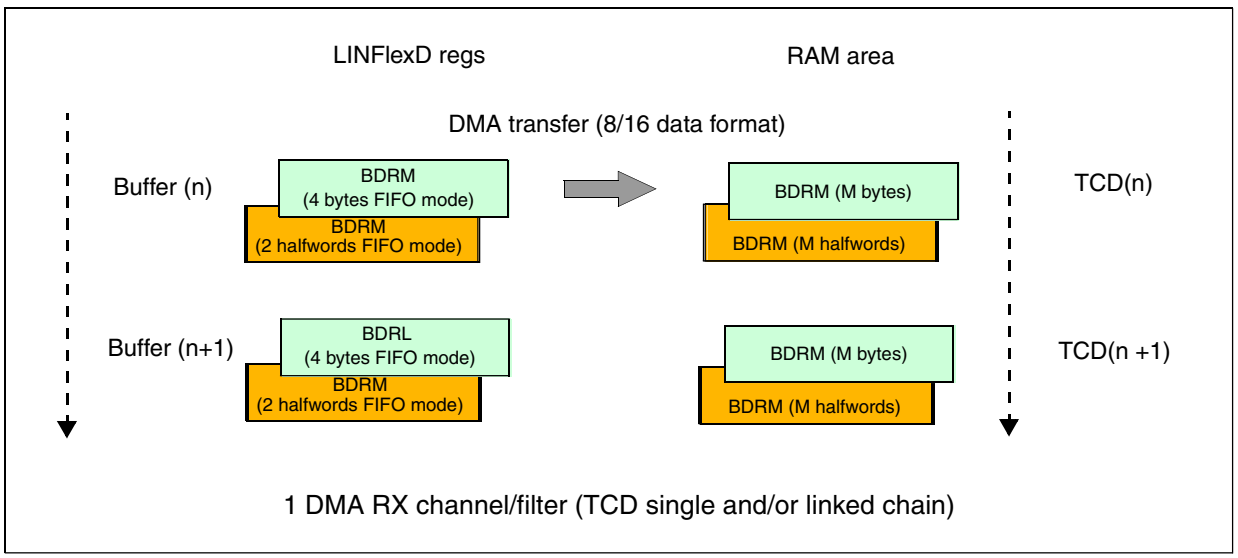


Figure 59-37. UART – RX memory map

The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format

This will sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

The concept FSM to control the DMA Rx interface is given in Figure 59-38. DMA Rx FSM will move to Idle state if DMARXE[0] = 0.The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/ halfword as soon an entry is available in the Rx FIFO. A new software reset bit is

required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 59-16. TCD setting – UART – Rx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	0		
DADDR[31:0]	-		RAM address
DOFF[15:0]	1	2	Byte/Half-word increment
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	–M	–M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

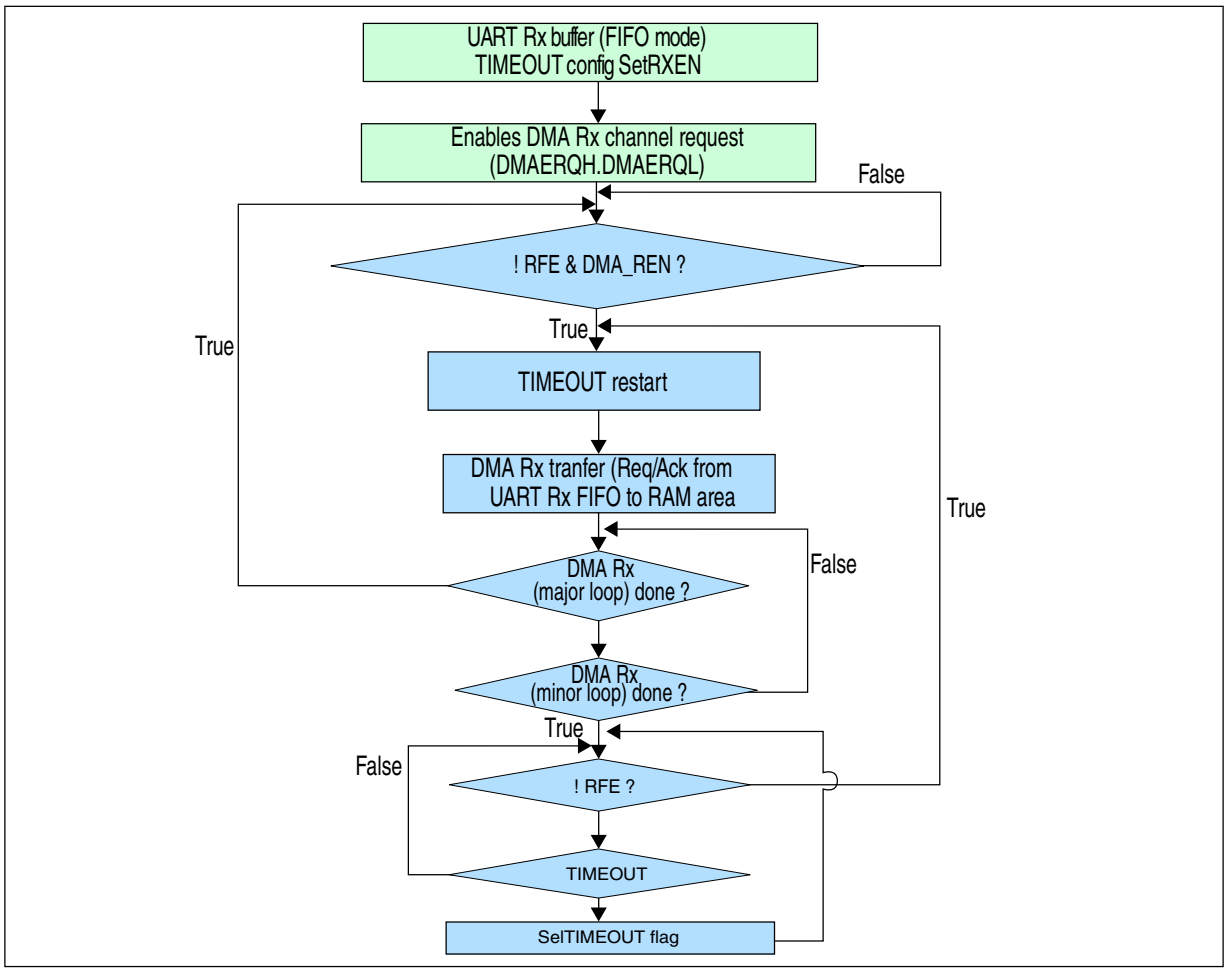


Figure 59-38. UART – DMA Rx FSM (concept scheme)

59.3.5.9 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index)
- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses

- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

59.4 Memory map and register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

NOTE

In Master mode, the registers IFCR0-IFCR15 are not present and the corresponding absolute address of the below register changes as shown in the following table.

Table 59-17. Offsets of the register from offsets 8C to 9C

Register name	Master/slave mode	Master only mode
LINFlexD_GCR	0x8C	0x4C
LINFlexD_UARTPTO	0x90	0x50
LINFlexD_UARTCTO	0x94	0x54
LINFlexD_DMATXE	0x98	0x58
LINFlexD_DMARXE	0x9C	0x5C

NOTE

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in ips_xfr_err being flagged.

LINFlexD memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LIN Control Register 1 (LINFlexD_LINCR1)	32	R/W	See section	59.4.1/3193
4	LIN Interrupt enable register (LINFlexD_LINIER)	32	R/W	0000_0000h	59.4.2/3197
8	LIN Status Register (LINFlexD_LINSR)	32	R/W	See section	59.4.3/3199
C	LIN Error Status Register (LINFlexD_LINESR)	32	w1c	0000_0000h	59.4.4/3203
10	UART Mode Control Register (LINFlexD_UARTCR)	32	R/W	0000_0000h	59.4.5/3204
14	UART Mode Status Register (LINFlexD_UARTSR)	32	R/W	0000_0000h	59.4.6/3210
18	LIN Time-Out Control Status Register (LINFlexD_LINTCSR)	32	R/W	0000_0200h	59.4.7/3213
1C	LIN Output Compare Register (LINFlexD_LINOCR)	32	R/W	0000_FFFFh	59.4.8/3214
20	LIN Time-Out Control Register (LINFlexD_LINTOCR)	32	R/W	See section	59.4.9/3215
24	LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)	32	R/W	0000_0000h	59.4.10/ 3215
28	LIN Integer Baud Rate Register (LINFlexD_LINIBRR)	32	R/W	0000_0000h	59.4.11/ 3216
2C	LIN Checksum Field Register (LINFlexD_LINCFR)	32	R/W	0000_0000h	59.4.12/ 3217
30	LIN Control Register 2 (LINFlexD_LINCR2)	32	R/W	See section	59.4.13/ 3218
34	Buffer Identifier Register (LINFlexD_BIDR)	32	R/W	0000_0000h	59.4.14/ 3220
38	Buffer Data Register Least Significant (LINFlexD_BDRL)	32	R/W	0000_0000h	59.4.15/ 3221
3C	Buffer Data Register Most Significant (LINFlexD_BDRM)	32	R/W	0000_0000h	59.4.16/ 3222
40	Identifier Filter Enable Register (LINFlexD_IFER)	32	R/W	0000_0000h	59.4.17/ 3222
44	Identifier Filter Match Index (LINFlexD_IFMI)	32	R	0000_0000h	59.4.18/ 3223
48	Identifier Filter Mode Register (LINFlexD_IFMR)	32	R/W	0000_0000h	59.4.19/ 3223
4C	Identifier Filter Control Register (LINFlexD_IFCR0)	32	R/W	0000_0000h	59.4.20/ 3224
50	Identifier Filter Control Register (LINFlexD_IFCR1)	32	R/W	0000_0000h	59.4.20/ 3224
54	Identifier Filter Control Register (LINFlexD_IFCR2)	32	R/W	0000_0000h	59.4.20/ 3224
58	Identifier Filter Control Register (LINFlexD_IFCR3)	32	R/W	0000_0000h	59.4.20/ 3224
5C	Identifier Filter Control Register (LINFlexD_IFCR4)	32	R/W	0000_0000h	59.4.20/ 3224
60	Identifier Filter Control Register (LINFlexD_IFCR5)	32	R/W	0000_0000h	59.4.20/ 3224
64	Identifier Filter Control Register (LINFlexD_IFCR6)	32	R/W	0000_0000h	59.4.20/ 3224

Table continues on the next page...

LINFlexD memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
68	Identifier Filter Control Register (LINFlexD_IFCR7)	32	R/W	0000_0000h	59.4.20/ 3224
6C	Identifier Filter Control Register (LINFlexD_IFCR8)	32	R/W	0000_0000h	59.4.20/ 3224
70	Identifier Filter Control Register (LINFlexD_IFCR9)	32	R/W	0000_0000h	59.4.20/ 3224
74	Identifier Filter Control Register (LINFlexD_IFCR10)	32	R/W	0000_0000h	59.4.20/ 3224
78	Identifier Filter Control Register (LINFlexD_IFCR11)	32	R/W	0000_0000h	59.4.20/ 3224
7C	Identifier Filter Control Register (LINFlexD_IFCR12)	32	R/W	0000_0000h	59.4.20/ 3224
80	Identifier Filter Control Register (LINFlexD_IFCR13)	32	R/W	0000_0000h	59.4.20/ 3224
84	Identifier Filter Control Register (LINFlexD_IFCR14)	32	R/W	0000_0000h	59.4.20/ 3224
88	Identifier Filter Control Register (LINFlexD_IFCR15)	32	R/W	0000_0000h	59.4.20/ 3224
8C	Global Control Register (LINFlexD_GCR)	32	R/W	0000_0000h	59.4.21/ 3225
90	UART Preset Timeout Register (LINFlexD_UARTPTO)	32	R/W	0000_0FFFh	59.4.22/ 3227
94	UART Current Timeout Register (LINFlexD_UARTCTO)	32	R/W	0000_0000h	59.4.23/ 3228
98	DMA Tx Enable Register (LINFlexD_DMATXE)	32	R/W	0000_0000h	59.4.24/ 3229
9C	DMA Rx Enable Register (LINFlexD_DMARXE)	32	R/W	0000_0000h	59.4.25/ 3229
A0	PSI5-S Tx Delay register (LINFlexD_PTD)	32	R/W	0000_0000h	59.4.26/ 3230

59.4.1 LIN Control Register 1 (LINFlexD_LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

NOTE

When accessing the LINCR1 register, each reserved bit should be written to its original reset value.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD	CFD	LASE	AUTOWU	MBL				BF	0	LBKM	MME	SSBL	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	*	0	0	1	0

* Notes:
- MME field: If slave = 0, then MME bit is 1, else MME bit is 0

LINFlexD_LINCR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CCD	Checksum Calculation disable This bit can be written during Initialization mode only. It is read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to sent a software calculated checksum/CRC (provided CFD is reset).
17 CFD	Checksum field disable This bit can be configured during Initialization mode only. It is read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame
18 LASE	LIN Autosynchronization Enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0). Register bit can be read in any mode, written only in initialization mode. 0 Autosynchronization disabled 1 Autosynchronization enabled
19 AUTOWU	Auto Wakeup This bit can be configured during Initialization mode only. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
	0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set
20–23 MBL	Master Break Length These bits choose the length of the Sync break to be generated by the master. These bits can be programmed in Initialization mode only. They are read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0000 10-bit break length 0001 11-bit break length 0010 12-bit break length 0011 13-bit break length 0100 14-bit break length 0101 15-bit break length 0110 16-bit break length 0111 17-bit break length 1000 18-bit break length 1001 19-bit break length 1010 20-bit break length 1011 21-bit break length 1100 22-bit break length 1101 23-bit break length 1110 36-bit (cooling) break length 1111 50-bit break length
24 BF	By-pass filter This bit can be programmed during Initialization mode only. **NOTE:** If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed. Register bit can be read in any mode, written only in initialization mode. 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBKM	Loop Back mode Refer to "Loop back mode" in Test mode . Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Loop Back Mode disabled 1 Loop Back mode enabled
27 MME	Master mode enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **NOTE:** If generic slave = 0, then this bit will read a '1' always, and cannot be programmed. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
	0 Slave Mode 1 Master Mode
28 SSBL	Slave Mode Sync Break Length This bit can be programmed in Initialization mode only. It is read only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 11 bit break length 1 10 bit break length
29 RBLM	Receiver Buffer Locked mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also. When PSI5S is not in UART_STDALONE mode, SLEEP bit can be set if it is in PS_DISABLE mode else this bit is cleared.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also. When PSI5S is not in UART_STDALONE mode, INIT bit can be set if it is in PS_CONFIG mode else this bit is cleared.

59.4.2 LIN Interrupt enable register (LINFlexD_LINIER)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0		FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIETOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINIER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZIE	Stuck at zero Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR. 0 No interrupt 1 Interrupt enabled
20 HEIE	Header Error Interrupt Enable An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **NOTE:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed.

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
	0 No interrupt 1 Interrupt enabled
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR. 0 No interrupt 1 Interrupt enabled
25 LSIE	LIN state Interrupt enable Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag. 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum
26 WUIE	Wakeup interrupt enable If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated. 0 No interrupt 1 Interrupt enabled
27 DBFIE	Data Buffer Full Interrupt enable An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set. 0 No interrupt 1 Interrupt enabled
28 DBEIETOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode). 0 No interrupt 1 Interrupt enabled
29 DRIE	Data Reception complete Interrupt enable An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
30 DTIE	Data Transmitted Interrupt enable An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
31 HRIE	Header Received Interrupt

Table continues on the next page...

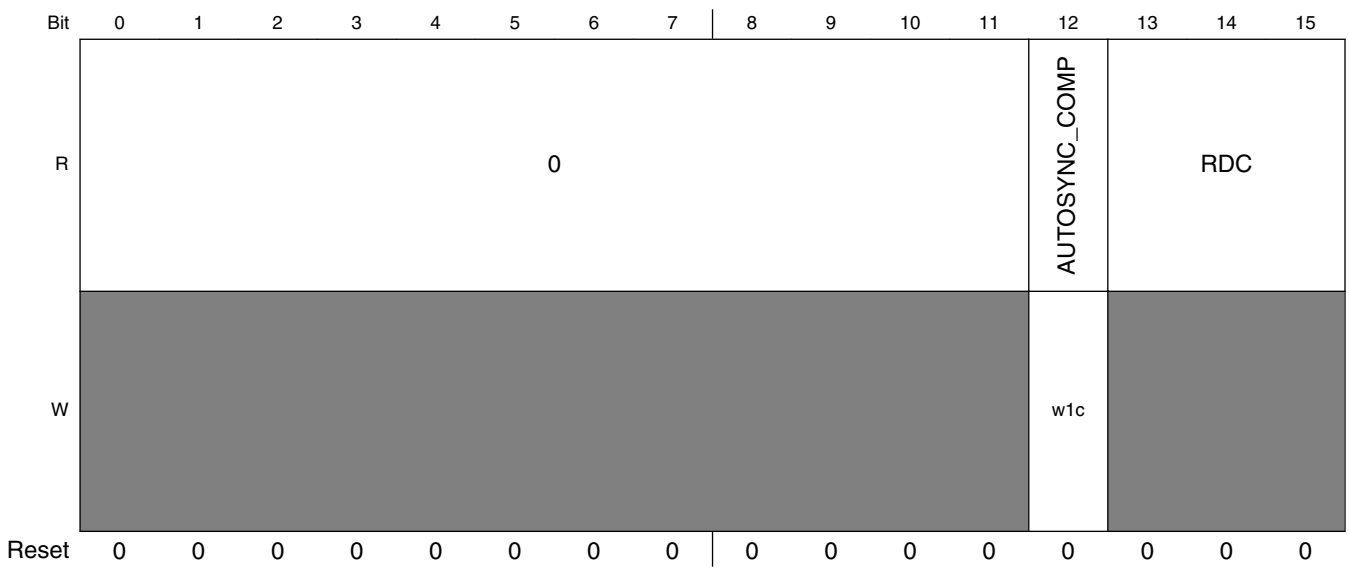
LINFlexD_LINIER field descriptions (continued)

Field	Description
	An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed. 0 No interrupt 1 Interrupt enabled

59.4.3 LIN Status Register (LINFlexD_LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0h base + 8h offset = 8h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0		RMB	DRBNE	RXbusy	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W	1						w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	*	0	0	0	0	0	0

* Notes:
- RDI field: Reset value of RDI reflects the RX pin state

LINFlexD_LINSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 AUTOSYNC_COMP	AUTOSYNC_COMP This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode. **NOTE:** If autosynchronization is disabled, this bit is reserved.
13–15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table continues on the next page...

LINFlexD_LINSR field descriptions (continued)

Field	Description
16–19 LINS	LIN state In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits. **NOTE:** The value of this bit field doesn't change by any write operation to it. Writing 0xF to this bit field clears the Rx interrupt, only when set due to the LIN state event. 0000 Sleep mode: LINFlexD is in Sleep mode, to save power consumption. 0001 Init mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0010 Idle mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0011 Sync break: In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. Note: In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0100 Sync Del: In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0101 Sync Field: In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0110 Identifier Field: In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0111 Header Reception/Transmission: In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 1000 Data Reception/Data Transmission: In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 1001 Checksum: Data transmission/reception completed, checksum transmission/reception ongoing.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXbusy	Receiver Busy flag In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit. 0 Receiver is idle 1 Reception ongoing

Table continues on the next page...

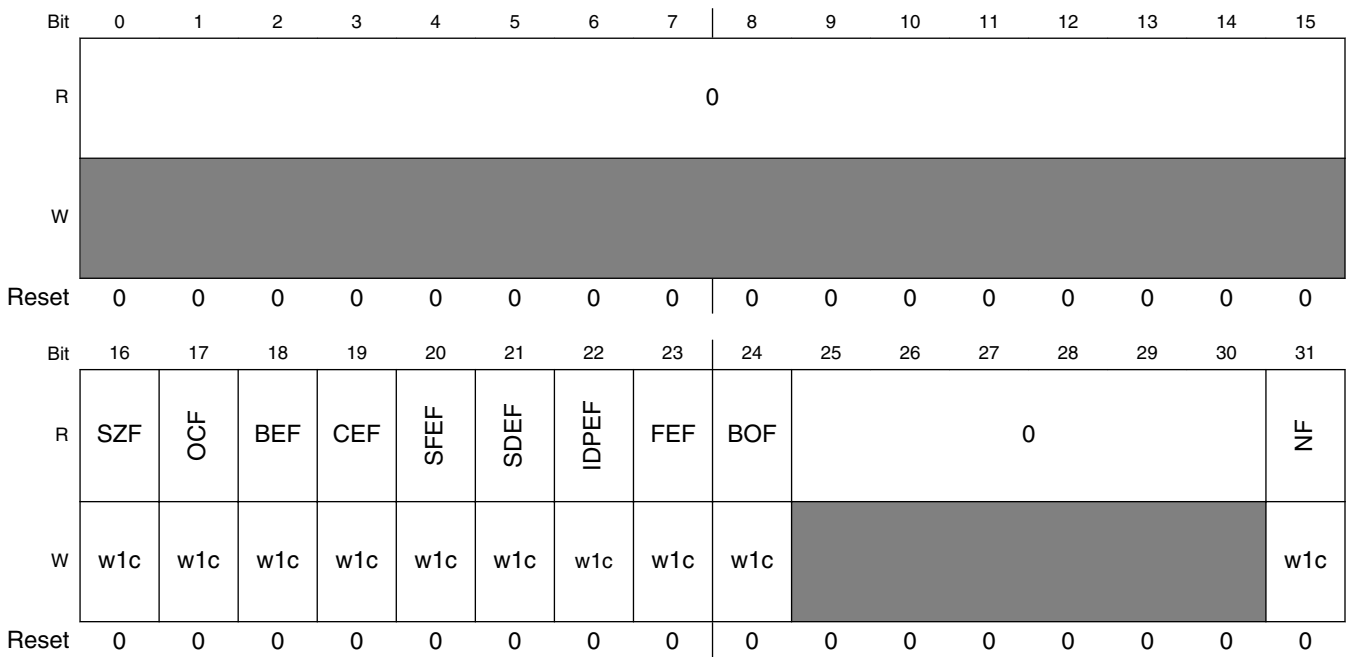
LINFlexD_LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **NOTE:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin. 1. When slave is in Sleep mode 2. When master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next 8 bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: a) All filters are inactive and Bypass filter (BF) bit is set b) No match in any filter and Bypass filter (BF) bit is set c) TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

59.4.4 LIN Error Status Register (LINFlexD_LINESR)

Refer to Errors for detailed description of all errors.

Address: 0h base + Ch offset = Ch



LINFlexD_LINESR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0: No output compare event occurred 1: In master mode, LINESR[OCF] flag is set when counter LINTCSR[CNT] has matched the content of LINOCR[OC2]. In slave mode, LINESR[OCF] is set when the content of the counter LINTCSR[CNT] has matched the content of LINOCR[OC1] or LINOCR[OC2]. This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.

Table continues on the next page...

LINFlexD_LINESR field descriptions (continued)

Field	Description
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **NOTE:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
25–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

59.4.5 UART Mode Control Register (LINFlexD_UARTCR)

NOTE

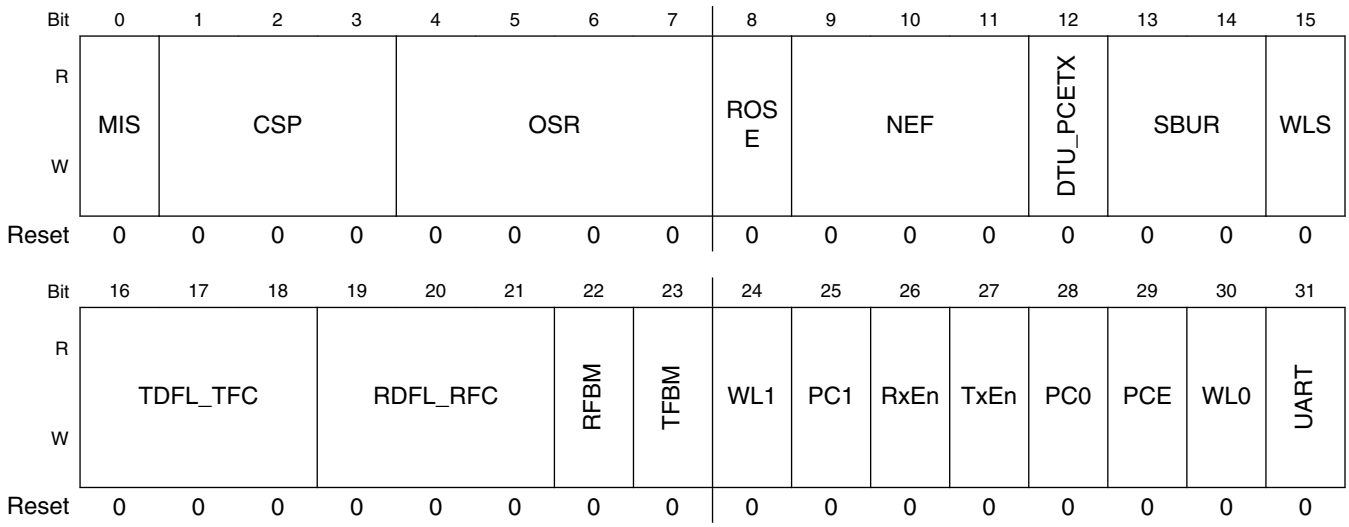
When accessing the UARTCR register, reserved bits should always be written to zero.

NOTE

The LINFlex module does not support communication with Special Word Length in UART mode set (UARTCR[WLS] = 1)

in buffer mode or in FIFO mode with UARTCR[WL1, WL0] = 0,1.

Address: 0h base + 10h offset = 10h



LINFlexD_UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State Register bit can be read in any mode, written only in initialization mode. 0 UARTCTO monitors the number of bits to be received. 1 UARTCTO monitors the idle state of the reception line.
1–3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling. CSP can take the following range of values for a certain over sampling rate: **OSR** / **CSP** 4 / 2,3 5 / 2, 3, 4 6 / 3, 4, 5 8 / NA Register bit can be read in any mode, written only in initialization mode.
4–7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled. Allowed values are: 4, 5, 6 and 8 . Register bit can be read in any mode, written only in initialization mode.
8 ROSE	Reduced Over Sampling EnableRegister bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate.
9–11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received. Register bit is a read/write field.
12 DTU_PCETX	Disable Timeout in UART mode/Parity transmission and checking This bit can be programmed in Initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received. **In Buffer mode:** This timer gets reset whenever the number of bytes received (rec_byte_cnt) is equal to RDFL. At this point the DRF bit is also set and num_of_frames (number of frames received), rec_byte_cnt are reset. When num_of_frames equals NEF, Disable Timeout is generated. But this clears num_of_frames also. Thus the timer restarts again. The value of counter is 0 when it restarts. **In FIFO mode:** When num_of_frames equals NEF, Disable Timeout is generated which clears num_of_frames and causes the timer to restart again. The value of counter is 0 when it restarts. Notes: • Disable Timeout causes only the Timer to restart which enables Timeout again. • Timer reset means it resets and starts counting again and the Timeout is enabled. When PSI5S is not in UART_STDALONE mode: 0 Parity transmission and checking disabled 1 Parity transmission and checking enabled When PSI5S is in UART_STDALONE mode: 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received
13–14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. This bit can be programmed in the initialization mode only, when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 reserved
15 WLS	Special Word Length in UART mode This bit can be programmed in initialization mode only when the UART bit is set. If this bit is set, setting WL and PCE bits has no effect in UART reception although parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	Register bit can be read in any mode, written only in Initialization mode. 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support.
16–18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 15 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes Register bit is a read/write field. TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TFCx is a read-only field available for debug purposes. 000 : empty 001 : 1 byte 010 : 2 bytes 011 : 3 bytes 100 : 4 bytes Others : reserved TDFLTFC can be programmed and are significative only when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. **NOTE:** When PSI5S is not in UART_STDALONE mode, these bits always reflects the value of PSI5S_GLCR[DIRCMD_LEN] **NOTE:** When a debugger is connected, this counter will decrement if TxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
19–21 RDFL_RFC	Reception Data Field Length /RX FIFO Counter RDFL defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations RDFL = x01 or RDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RFCx is a read-only field available for debug purposes. 000 : Empty 001 : 1 byte

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	010 : 2 bytes/1.5 byte in case of WLS bit setting 011 : 3 bytes 100 : 4 bytes/2x1.5 byte in case of WLS bit setting Others - Reserved RDFLRFC can be programmed and are significative only when the UART bit is set. **NOTE:** In buffer mode, RDFL should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint. **NOTE:** When a debugger is connected, this counter will decrement if RxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
22 RFBM	RFBM Rx Fifo/Buffer mode This bit can be programmed in Initialization mode, only when the UART bit is set. Register bit can be read in any mode, written only in initialization mode. 0 Rx Buffer mode enabled 1 Rx Fifo mode enabled (mandatory in DMA Rx mode)
23 TFBM	Tx Fifo/Buffer mode This bit can be programmed in initialization mode, only when the UART bit is set. Register bit can be read in any mode, written only in initialization mode. 0 Tx Buffer mode enabled 1 Tx Fifo mode enabled (mandatory in DMA Tx mode)
24 WL1	Word Length in UART mode This bit can be programmed in Initialization mode only, when the UART bit is set. Register bit can be read in any mode, written only in initialization mode.

WL1	WL0	Description
0	0	7 bits data + parity
0	1	8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1
1	0	15 bits data + parity
1	1	16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1

25 PC1	Parity Control This bit can be programmed in Initialization mode, only when UART bit is set. Register bit can be read in any mode, written only in initialization mode.

PC1	PC0	Description
0	0	Parity sent is Even
0	1	Parity sent is Odd

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>26
RxEn</td><td>Receiver Enable
This bit can be programmed only when the UART bit is set.
0 Receiver disabled
1 Receiver enabled</td></tr>
<tr><td>27
TxEn</td><td>Transmitter Enable
This bit can be programmed only when UART bit is set.
0 Transmitter disabled
1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed</td></tr>
<tr><td>28
PC0</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>29
PCE</td><td>Parity Control Enable
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Parity transmit/check Disable
1 Parity transmit/check Enable</td></tr>
<tr><td>30
WL0</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
<tr><td>0</td><td>1</td><td>8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description		
	WL1	**WL0**	**Description**
	1	0	15 bits data + parity
	1	1	16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1
31 UART	UART Mode This bit can be programmed in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 LIN mode 1 UART mode		

59.4.6 UART Mode Status Register (LINFlexD_UARTSR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRFRFE	DTFTFF	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. **NOTE:** For Baud rate above 1 Mbit/s, this flag is not usable. 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR
18–21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **NOTE:** Parity is checked only after complete frame is received. Following are the conditions when WL bits = 01 or 11 (either PE0 or PE2 is only set). • When PE0 is set, it indicates Parity error in either first or second byte received • When PE2 is set, it indicates Parity error in either third or fourth byte received 0 No parity error 1 Parity error in the corresponding received byte
22 RMB	Release Message Buffer This bit should be cleared by software. This bit will reflect the same value as in LINSR, when in Initialization mode and the UART bit is set. 0 Buffer data is free 1 Buffer data ready to be read by software
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the new value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.

Table continues on the next page...

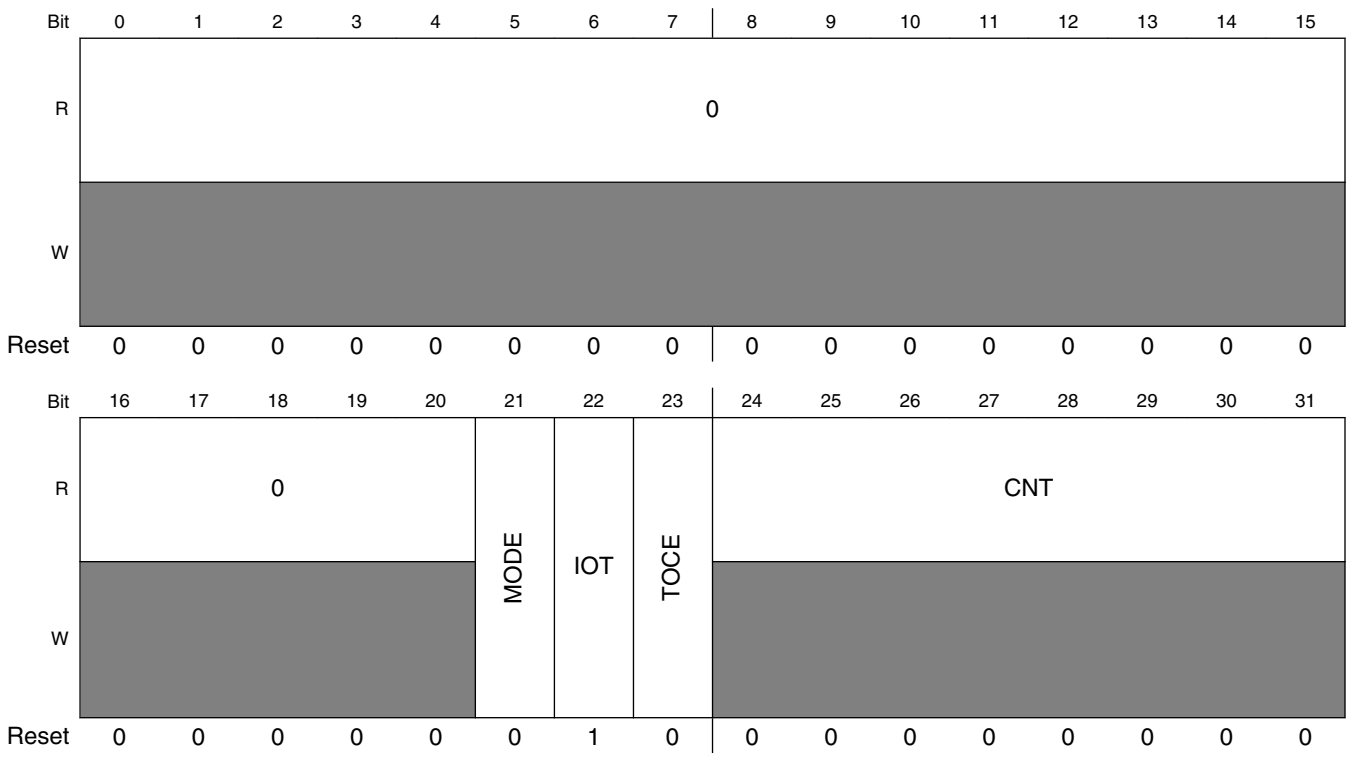
LINFlexD_UARTSR field descriptions (continued)

Field	Description
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software. The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRFRFE	Data Reception Completed Flag /Rx FIFO Empty Flag DRF is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF bit is set when the configured number of valid stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF is set irrespective of framing error in case framing error is in the last STOP bit configured, parity error or overrun error. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTFTFF	Data Transmission Completed Flag/ TX FIFO Full Flag DTF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

59.4.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)

This register contains control and status bits for timeout feature.

Address: 0h base + 18h offset = 18h



LINFlexD_LINTCSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MODE	Time-out counter mode This bit can be configured only during initialization. Register bit can be read in any mode, written only in initialization mode. 0 LIN mode 1 Output compare mode
22 IOT	Idle on timeout Register bit can be read in any mode, written only in initialization mode. This feature is applicable only when MODE bit in LINTCSR is cleared. 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event

Table continues on the next page...

LINFlexD_LINTCSR field descriptions (continued)

Field	Description
23 TOCE	Time-out counter enable TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE. 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs.
24–31 CNT	Counter Value These bits reflect the value of a free-running counter used for timeout. **NOTE:** For proper functionality of this counter, LINIBRR should be >= 5.

59.4.8 LIN Output Compare Register (LINFlexD_LINOCR)

This register contains the value to be compared to the LINTCSR:CNT value. This register is writable by software only in Output Compare Mode.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																OC2								OC1							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_LINOCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 OC2	Output compare value 2
24–31 OC1	Output compare value 1

59.4.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	*	*	*	*	*	*	*

* Notes:

- HTO field: HTO reset values:

  HTO resets to 0011100b in Master only mode, if generic slave = 0.

  HTO resets to 0101100b in Master/Slave mode, if generic slave = 1.

LINFlexD_LINTOCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to $T_{Response\_Maximum} = 1.4 \times T_{Response\_Nominal}$
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. If Master mode is enabled then these bits will always reflect 28 (1.4 x 10 bits of sync + 1.4 x 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

59.4.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

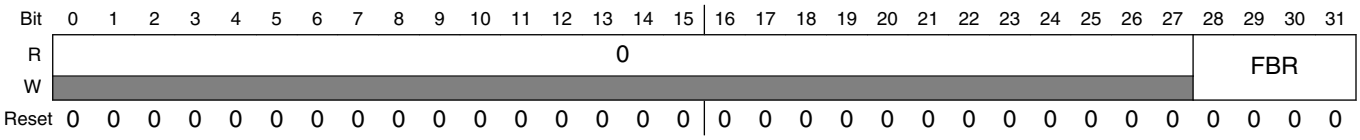
NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

NOTE

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0h base + 24h offset = 24h



LINFlexD_LINFBRR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 FBR	Fractional Baud rates Register bit can be read in any mode, written only in initialization mode. 0000 Fraction(LDIV) = 0 0001 Fraction(LDIV) = 1/16 0010 Fraction(LDIV) = 2/16 0011 Fraction(LDIV) = 3/16 0100 Fraction(LDIV) = 4/16 0101 Fraction(LDIV) = 5/16 0110 Fraction(LDIV) = 6/16 0111 Fraction(LDIV) = 7/16 1000 Fraction(LDIV) = 8/16 1001 Fraction(LDIV) = 9/16 1010 Fraction(LDIV) = 10/16 1011 Fraction(LDIV) = 11/16 1100 Fraction(LDIV) = 12/16 1101 Fraction(LDIV) = 13/16 1110 Fraction(LDIV) = 14/16 1111 Fraction(LDIV) = 15/16

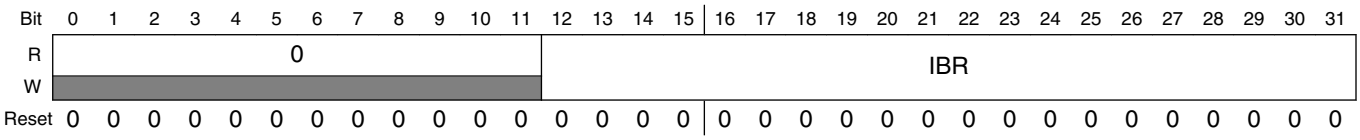
59.4.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0h base + 28h offset = 28h



LINFlexD_LINIBRR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. IBR = 0h: LIN clock disabled IBR = 1h: Mantissa (LDIV) = 1 ... IBR = FFFFEh: Mantissa (LDIV) = 1048574 IBR = FFFFFh: Mantissa (LDIV) = 1048575 Register bit can be read in any mode, written only in initialization mode.

59.4.12 LIN Checksum Field Register (LINFlexD_LINCFR)

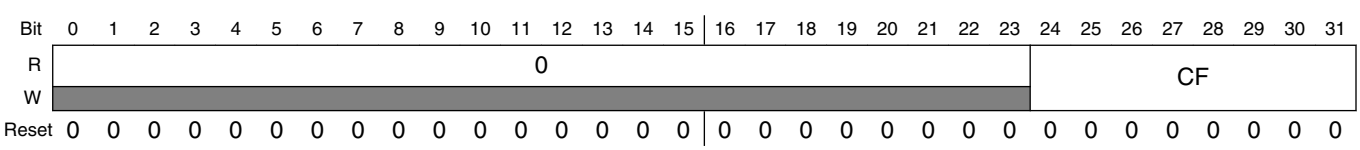
NOTE

There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.

This register consists of checksum bits.

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	1	read/write	None
1	0	read-only	None
0	1	read/write	Programmed checksum
0	0	read-only	Calculated checksum

Address: 0h base + 2Ch offset = 2Ch



LINFlexD_LINCFR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 CF	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software.

59.4.13 LIN Control Register 2 (LINFlexD_LINCR2)

This register includes control status bits related to buffer operations.

NOTE

When accessing the LINCR2 register, each reserved bit should be written to its original reset value.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE	IOBE	IOPE	WURQ	DDRQ	DTRQ	ABRQ	HTRQ	0							
W																
Reset	0	1	*	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- IOPE field: When slave = 0, this field always reads ‘0’ and cannot be programmed else this bit is programmable and reset value is 1.

LINFlexD_LINCR2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TBDE	Two Bit delimiter bit This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits
17 IOBE	Idle on Bit Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine
18 IOPE	Idle on Identifier Parity Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. If generic slave = 0, then this bit will always read a 1 else 0, and cannot be programmed. 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request. Register bit can be read/set by software
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered. Register bit can be read/set by software
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete. Register bit can be read/set by software
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also. Register bit can be read/set by software Register bit can be read/set by software
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **NOTE:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.4.14 Buffer Identifier Register (LINFlexD_BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

NOTE

All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BIDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL = Number of data bytes - 1 **NOTE:** DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the data from the BDR registers

Table continues on the next page...

LINFlexD_BIDR field descriptions (continued)

Field	Description
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data filed only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = '1').

59.4.15 Buffer Data Register Least Significant (LINFlexD_BDRL)

This register is a part of an 8-byte data buffer.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA3								DATA2								DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRL field descriptions

Field	Description
0–7 DATA3	Data Byte 3 Data byte 3 of the data field.
8–15 DATA2	Data Byte 2 Data byte 2 of the data field.
16–23 DATA1	Data Byte 1 Data byte 1of the data field.
24–31 DATA0	Data Byte 0 Data byte 0 of the data field.

59.4.16 Buffer Data Register Most Significant (LINFlexD_BDRM)

This register is a part of an 8-byte data buffer.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA7								DATA6								DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRM field descriptions

Field	Description
0–7 DATA7	Data Byte 7 Data byte 7 of the data field.
8–15 DATA6	Data Byte 6 Data byte 6 of the data field.
16–23 DATA5	Data Byte 5 Data byte 5of the data field.
24–31 DATA4	Data Byte 4 Data byte 4 of the data field.

59.4.17 Identifier Filter Enable Register (LINFlexD_IFER)

This register enables/disables a particular filter.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																FACT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

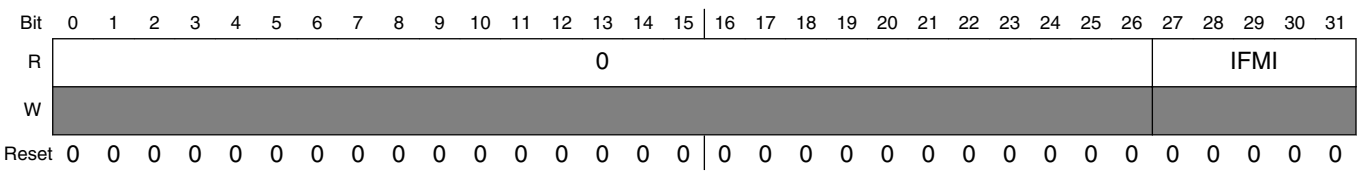
LINFlexD_IFER field descriptions (continued)

Field	Description
	1. In Identifier mask mode, FACT (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **NOTE:** Refer to chip configuration details to see the number of filters used in the device.

59.4.18 Identifier Filter Match Index (LINFlexD_IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0h base + 44h offset = 44h



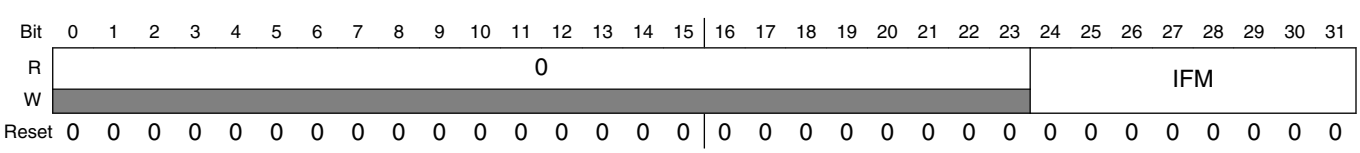
LINFlexD_IFMI field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Slave mode for more details).

59.4.19 Identifier Filter Mode Register (LINFlexD_IFMR)

This register configures the modes of filters.

Address: 0h base + 48h offset = 48h



LINFlexD_IFMR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 IFM	Filter mode Register bit can be read in any mode, written only in initialization mode. 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

59.4.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode.

Even-numbered instances (IFCR0, IFCR2, IFCR4, ...) of this register act as filters in both Identifier list and Identifier mask modes.

Odd-numbered instances (IFCR1, IFCR3, IFCR5, ...) of this register act:

- As filters in Identifier list mode
- As a mask for the preceding-numbered register in identifier mask mode

Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFCR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1 Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_IFCR*n* field descriptions (continued)

Field	Description
22 DIR	Direction This bit controls the direction of the data field. Register bit can be read in any mode, written only in initialization mode. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. Register bit can be read in any mode, written only in initialization mode. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. Register bit can be read in any mode, written only in initialization mode.

59.4.21 Global Control Register (LINFlexD_GCR)

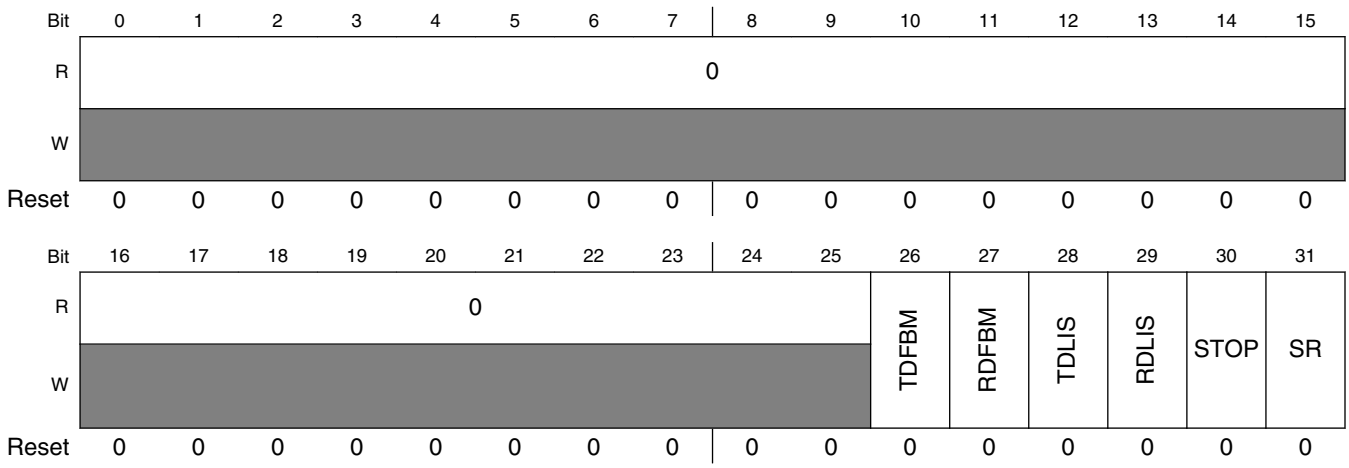
This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

The address offset depends on the no_of_filters. Refer to the chip configuration details for the number of filters used in this device.

Table 59-18. Register fields reset by SR

Register	Comment
LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset
LINESR	All fields are reset
LINTCSR	Only CNT[0:7] is reset
UARTSR	All fields except RFE & TFF are reset
UARTCR	Only TFC & RFC are reset
UARTCTO	All fields are reset

Address: 0h base + 8Ch offset = 8Ch



LINFlexD_GCR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Received data is not inverted 1 Received data is inverted

Table continues on the next page...

LINFlexD_GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). Register bit can be read in any mode, written only in initialization mode. 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). Register bit can be written only be software in initialization mode. Bit is always read 0 by software. Table 59-18 describes the register fields reset by SR.

59.4.22 UART Preset Timeout Register (LINFlexD_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be receivedor to monitor the idle state of the reception line. This register can be written by software any time.

The address offset depends on no_of_filters. Refer to the chip configuration details to see the number of filters used in the device.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																PTO															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_UARTPTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer also to register UARTCTO.

59.4.23 UART Current Timeout Register (LINFlexD_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see UART Preset Timeout Register (LINFlexD_UARTPTO) to monitor the number of bits received by UARTor to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

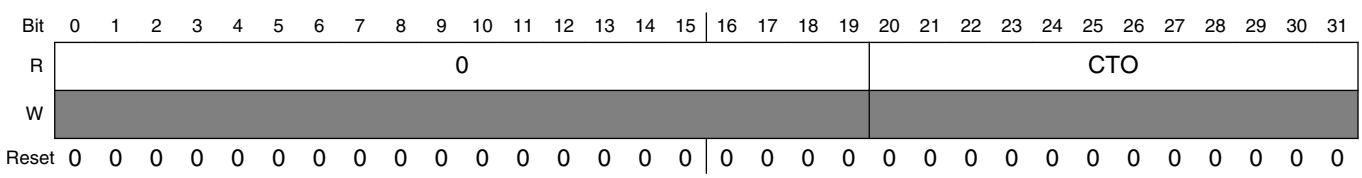
- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 94h offset = 94h



LINFlexD_UARTCTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 CTO	Current Timeout

Table continues on the next page...

LINFlexD_UARTCTO field descriptions (continued)

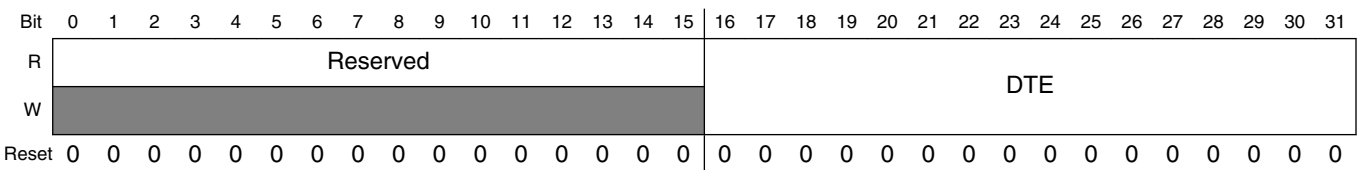
Field	Description
	CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

59.4.24 DMA Tx Enable Register (LINFlexD_DMATXE)

This register enables the DMA TX interface. This register can be written and read by software anytime.

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 98h offset = 98h



LINFlexD_DMATXE field descriptions

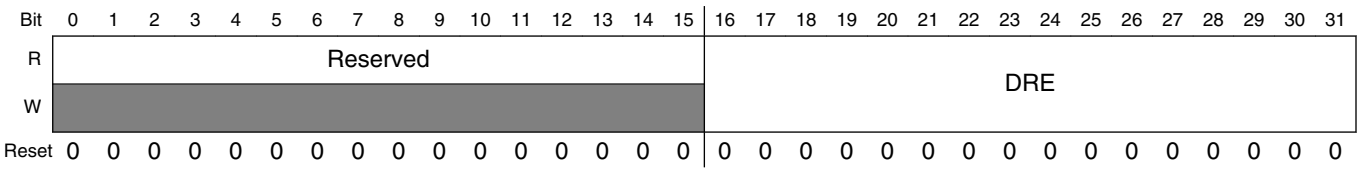
Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{TX\_CH\_NUM}$. These lies from 0 to (31 - $2^{TX\_CH\_NUM}$). Refer to the chip configuration details for the value of TX_CH_NUM used in this device.
16–31 DTE	DMA Tx channel Y enable **NOTE:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. The size is [$2^{TX\_CH\_NUM}$ - 1 : 0]. The position of these bits are (32 - $2^{TX\_CH\_NUM}$) to 31. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of TX_CH_NUM used in this device. 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled

59.4.25 DMA Rx Enable Register (LINFlexD_DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

The address offset depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Address: 0h base + 9Ch offset = 9Ch



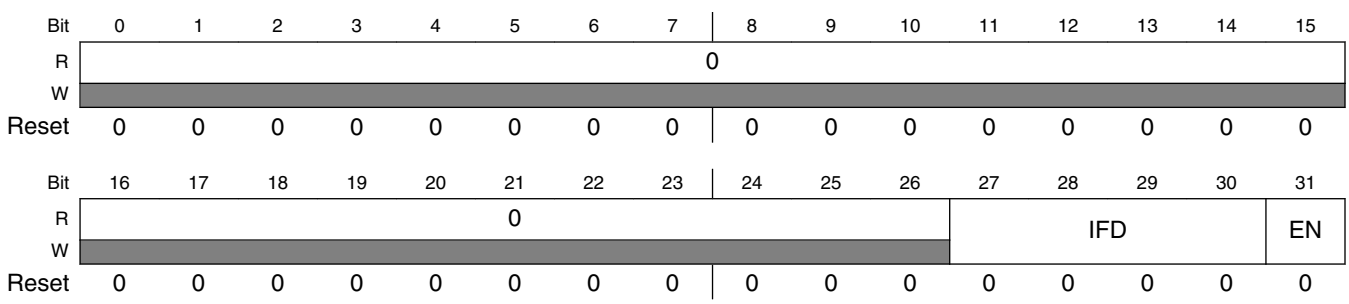
LINFlexD_DMARXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{RX\_CH\_NUM}$. These lies from 0 to (31 - $2^{RX\_CH\_NUM}$). Refer to the chip configuration details for the value of RX_CH_NUM used in this device.
16–31 DRE	DMA Rx channel Y enable **NOTE:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. The size is [$2^{RX\_CH\_NUM}$ -1 : 0]. The position of these bits are (32 - $2^{RX\_CH\_NUM}$) to 31. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of RX_CH_NUM used in this device. 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled

59.4.26 PSI5-S Tx Delay register (LINFlexD_PTD)

This register can be used to delay the generation of data transmision complete flag by a duration between 1 to 16 bit times.

Address: 0h base + A0h offset = A0h



LINFlexD_PTD field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 IFD	Interframe Delay The DTF generation is delayed by IFD+1 bit times. **NOTE:** If generic psi5 = 0, these bits always read 0 and cannot be programmed.
31 EN	Enable **NOTE:** If generic psi5 = 0, this bit always reads 0 and cannot be programmed. 0 DTF generation is not delayed 1 DTF generation is delayed based on IFD field

59.5 Programming considerations

The next subsections describe the various configurations in which the LINFlexD module can be used.

59.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

59.5.1.1 Transmitter

Transmitter Master Node - Transmitter configuration.

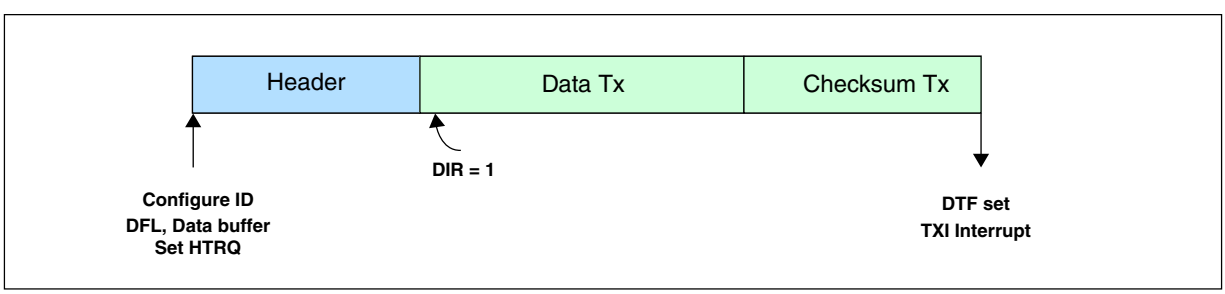


Figure 59-39. Master node – Transmitter configuration

59.5.1.2 Receiver

Receiver Master Mode - RX Configuration

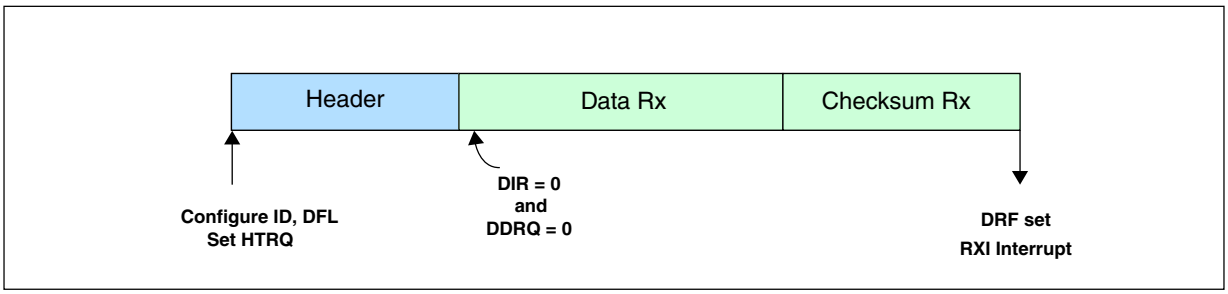


Figure 59-40. Master node – Receiver configuration

59.5.1.3 Transmitter, Bit Error

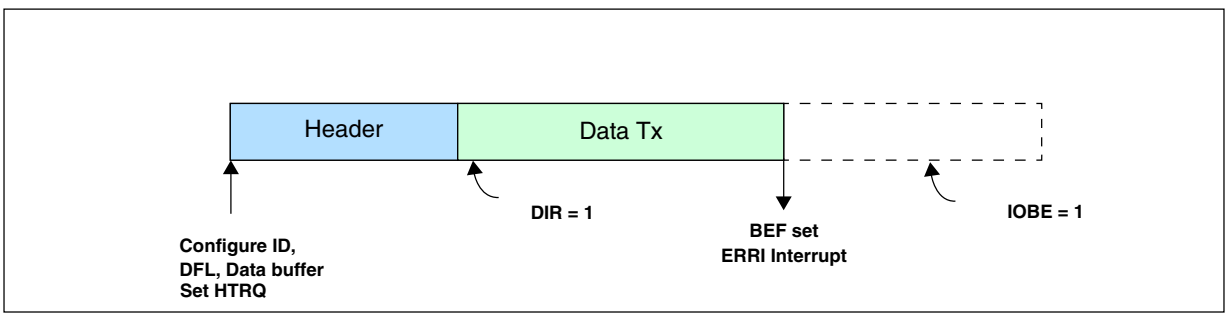


Figure 59-41. Master node – Transmitter, Bit Error configuration

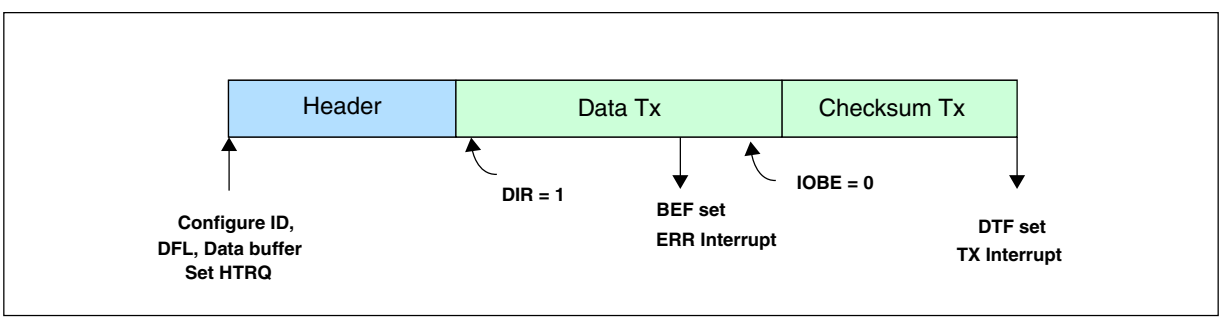


Figure 59-42. Master node – Transmitter, Checksum Error configuration

59.5.1.4 Receiver, Checksum Error

Checksum Error for Receiver.

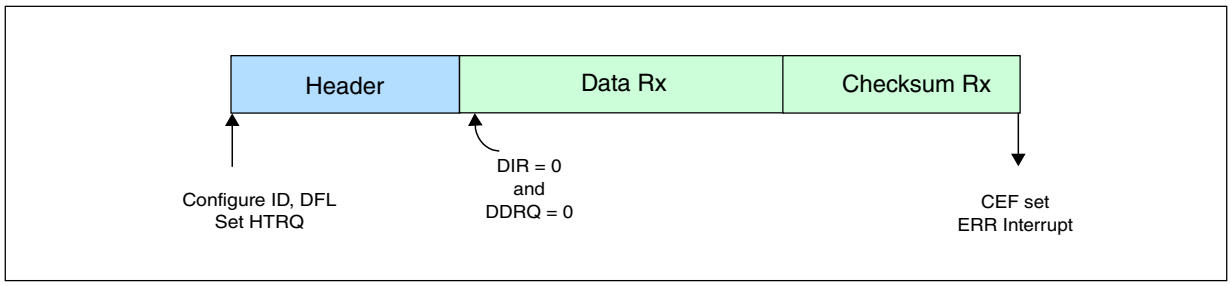


Figure 59-43. Master node – Receiver, Checksum Error configuration

59.5.2 Slave node

Slave node (transmitter).

59.5.2.1 Transmitter (no identifier filters)

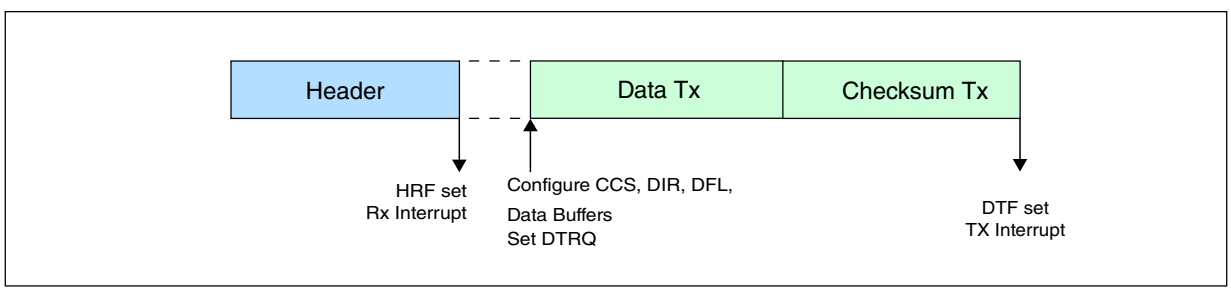


Figure 59-44. Slave node – Transmitter (no identifier filters) configuration

59.5.2.2 Receiver (no identifier filters)

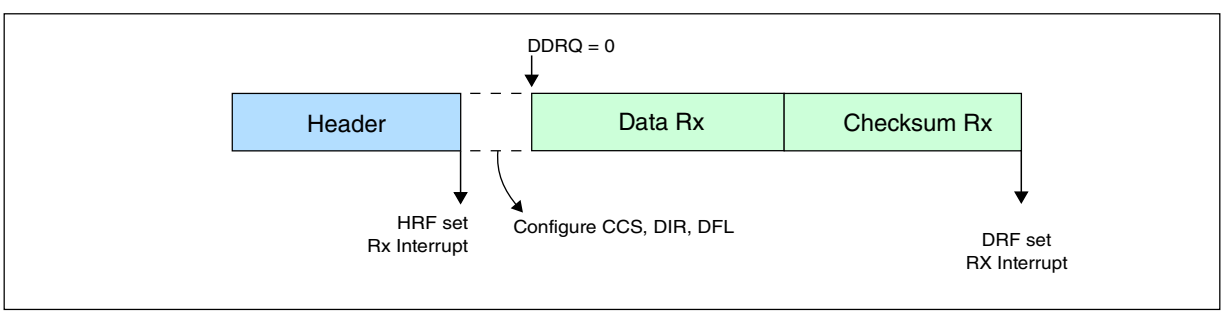


Figure 59-45. Slave node – Receiver (no identifier filters) configuration

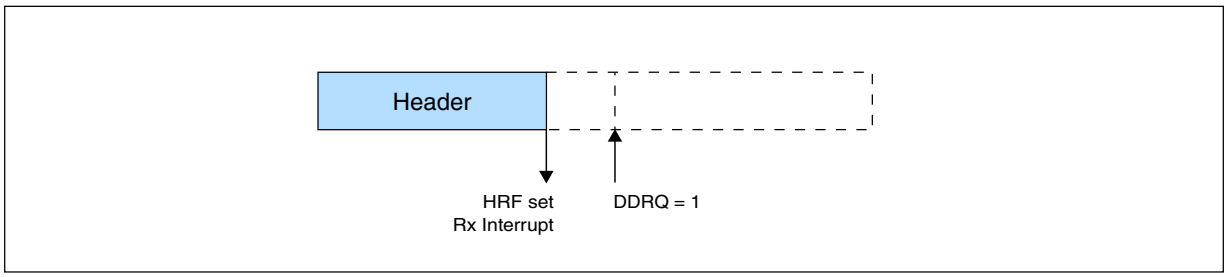


Figure 59-46. Slave node – Receiver (no identifier filters) configuration

59.5.2.3 No filters, Transmitter, Bit error

The following figure shows Slave node - No filters, Transmitter, Bit error configuration.

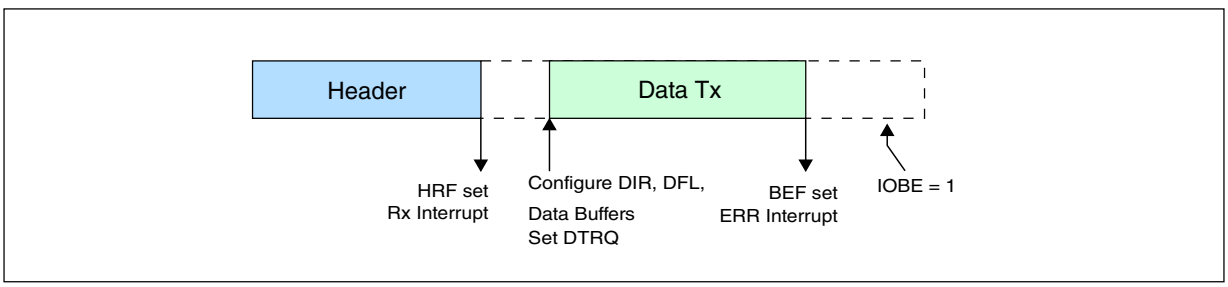


Figure 59-47. Slave node – No filters, Transmitter, Bit error configuration

59.5.2.4 No filters, Receiver, Checksum Error

The following figure shows Slave node - No filters, Receiver, Checksum error configuration.

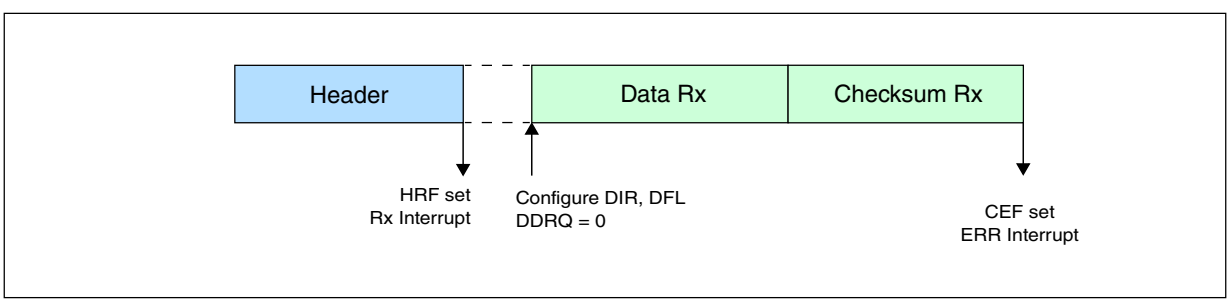


Figure 59-48. Slave node – No filters, Receiver, Checksum error configuration

59.5.2.5 At least one TX filter, BF is reset, ID matches filter

This configuration can be used in case slave never receives data, for example, sensor.

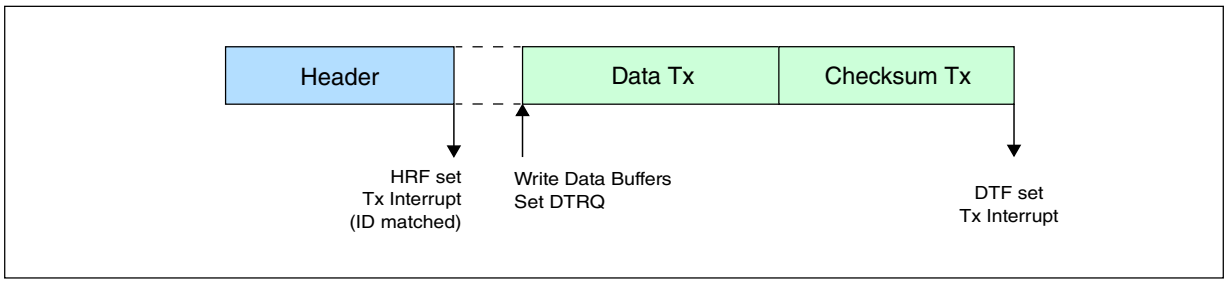


Figure 59-49. Slave node – one TX filter configuration

59.5.2.6 At least one RX filter, BF reset, ID matches filter

The following figure shows Slave node - one RX filter configuration.

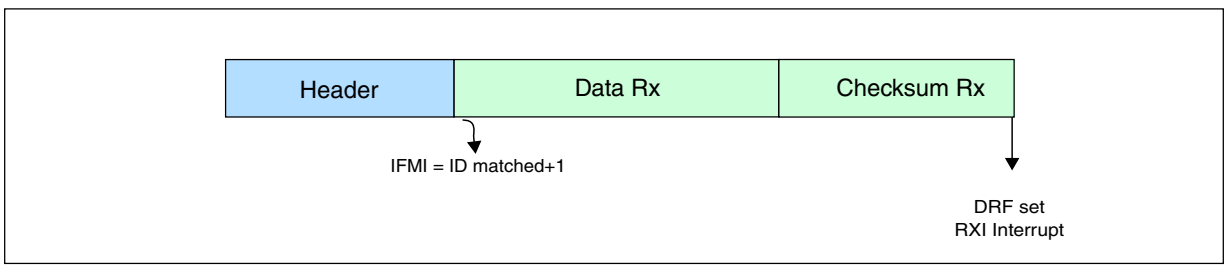


Figure 59-50. Slave node – one RX filter configuration

59.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

The following figure shows RX only, TX only, RX and TX filters configuration.

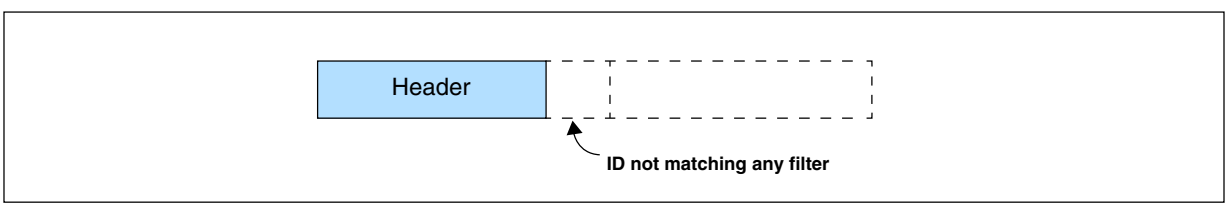


Figure 59-51. RX only, TX only, RX & TX filters configuration

59.5.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

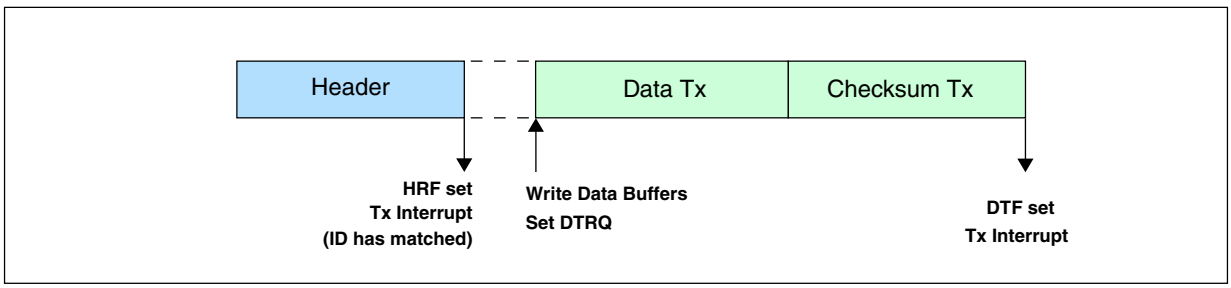


Figure 59-52. TX filter, BF is set configuration – ID has matched

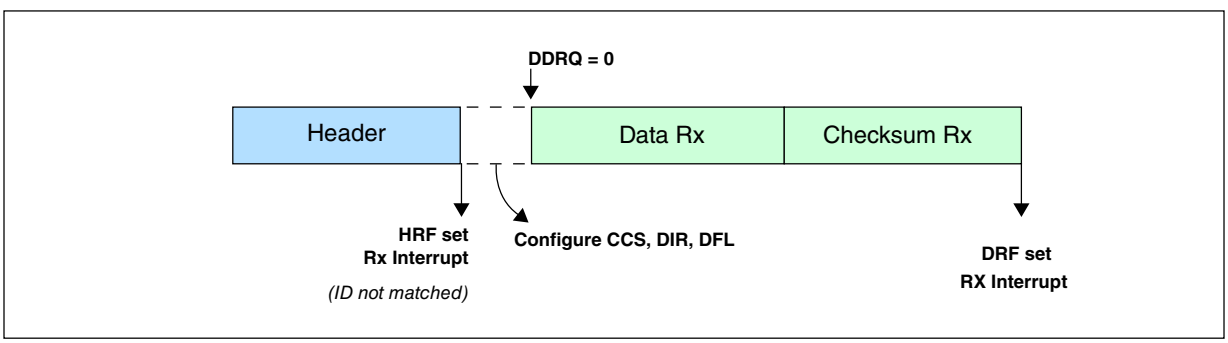


Figure 59-53. TX filter, BF is set configuration – ID not matched

59.5.2.9 RX filter, BF is set

The following figures show RX filter configurations.

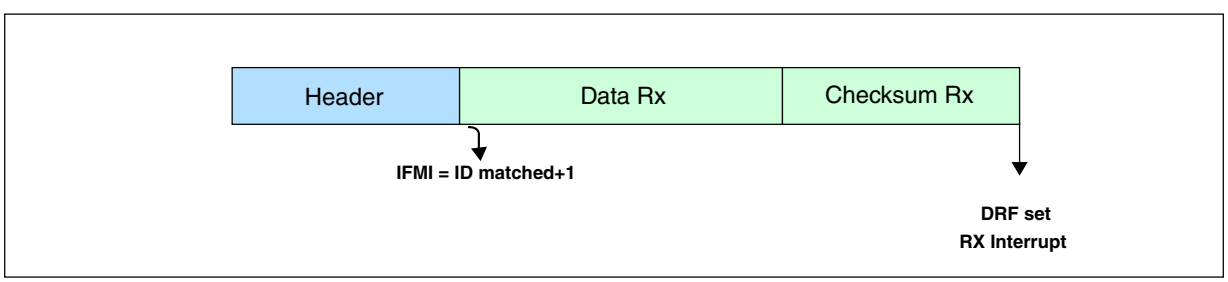


Figure 59-54. RX filter, BF is set configuration – ID matched

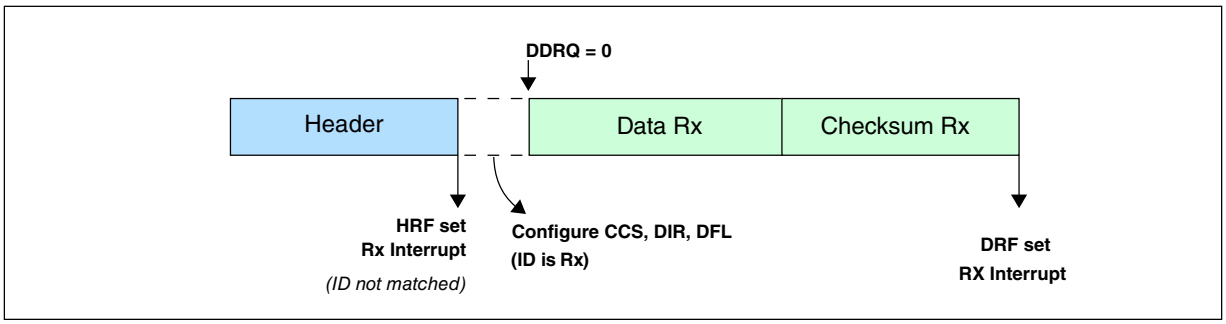


Figure 59-55. RX filter, BF is set configuration – ID not matched (ID is Rx)

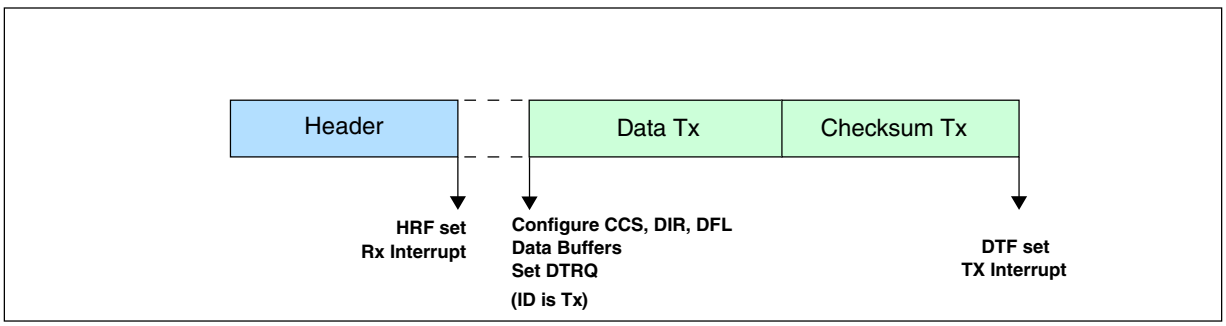


Figure 59-56. RX filter, BF is set configuration – (ID is Tx)

59.5.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

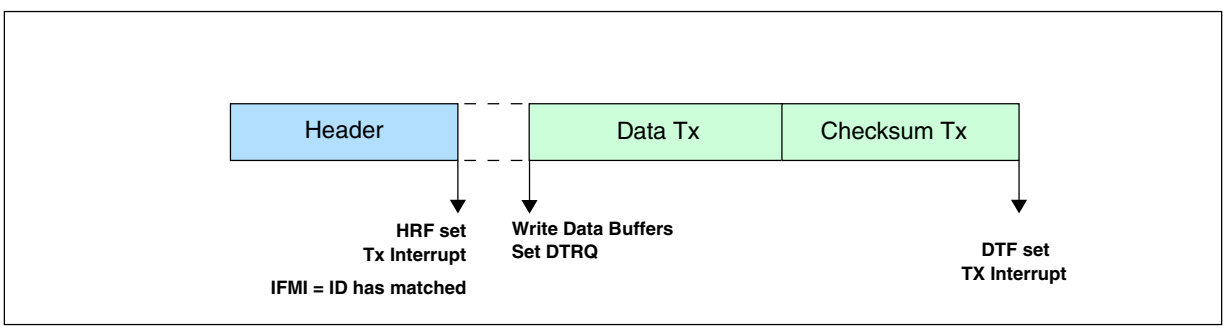


Figure 59-57. TX filter, RX filter, BF set configuration – (ID matched)

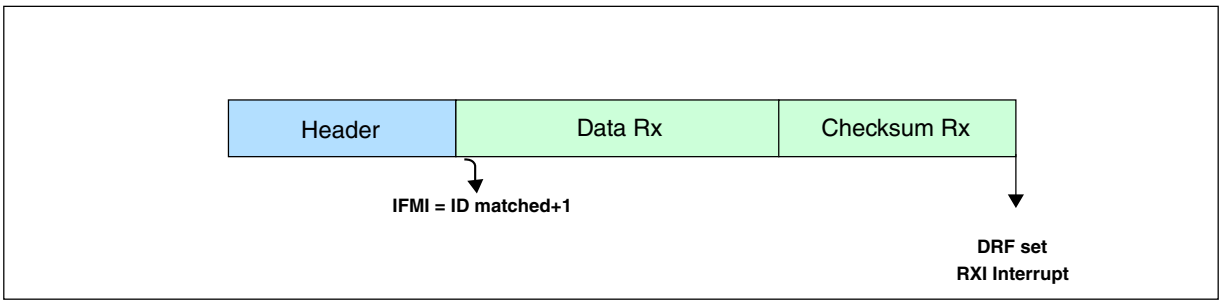


Figure 59-58. TX filter, RX filter, BF set configuration – (ID matched + 1)

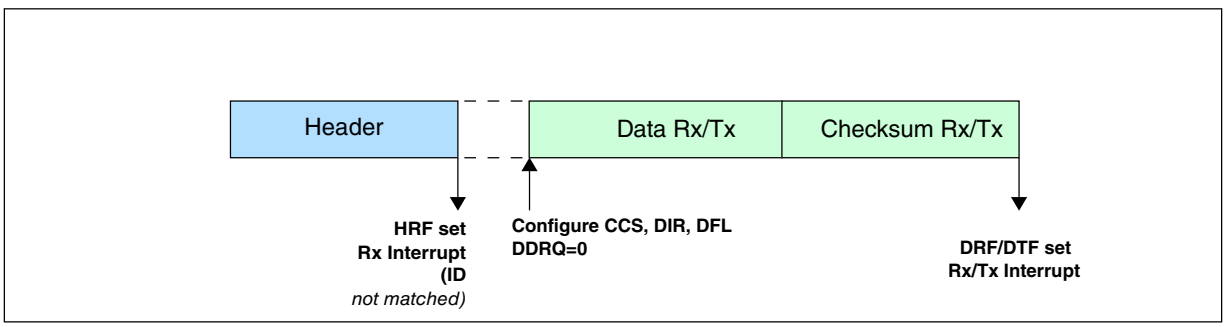


Figure 59-59. TX filter, RX filter, BF set configuration – (ID not matched, RX/TX Interrupt)

59.5.3 Extended frames

The following figure shows Extended frames (RX Interrupt).

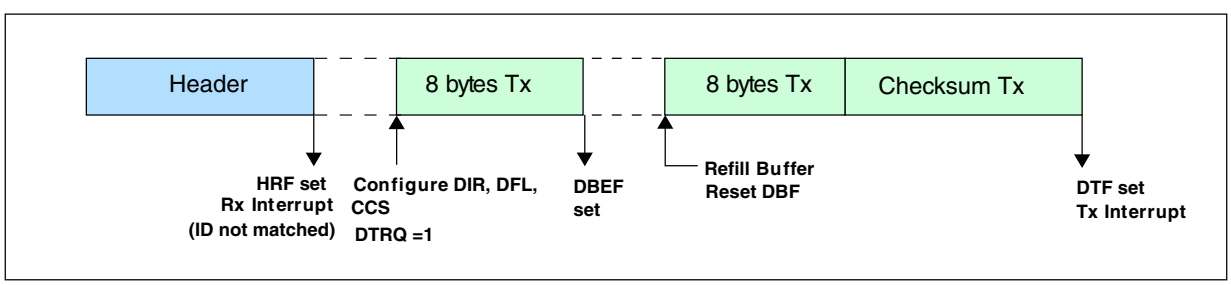


Figure 59-60. Extended frames (TX Interrupt)

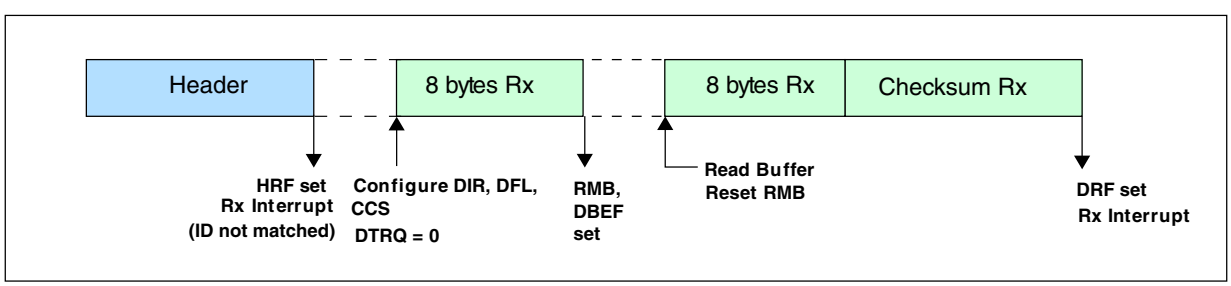


Figure 59-61. Extended frames (RX Interrupt)

59.5.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

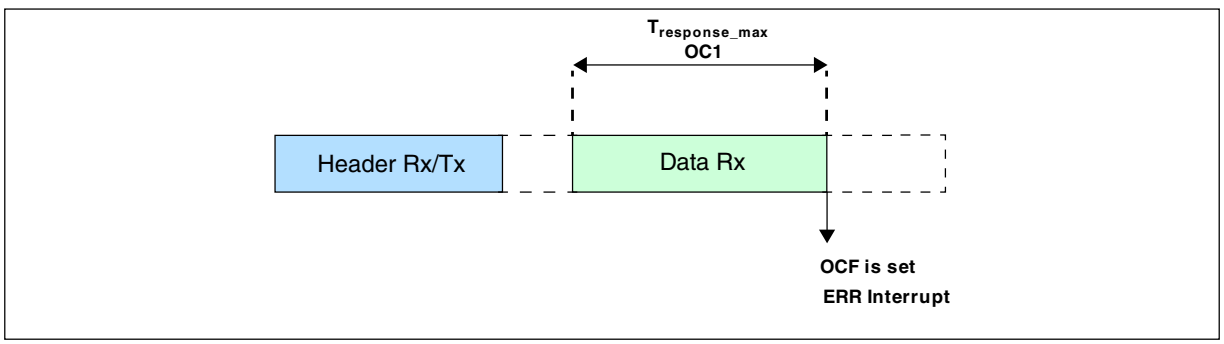


Figure 59-62. Response timeout

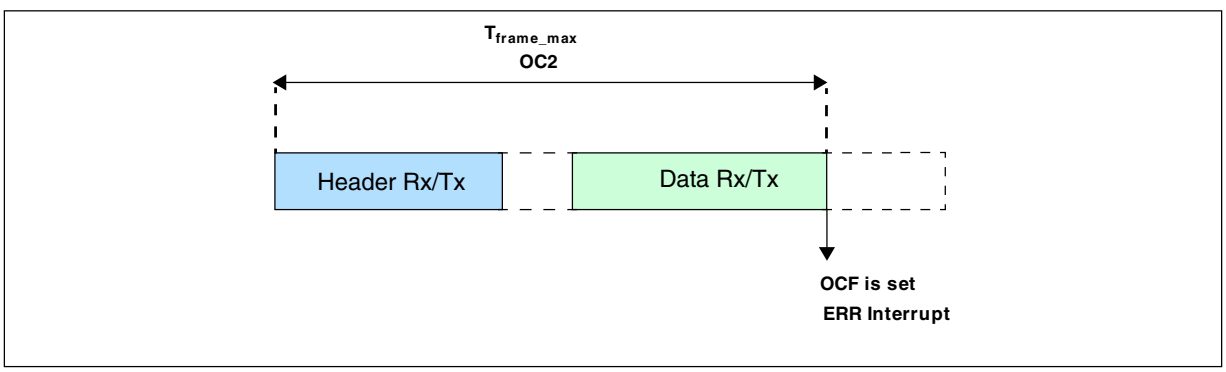


Figure 59-63. Frame timeout

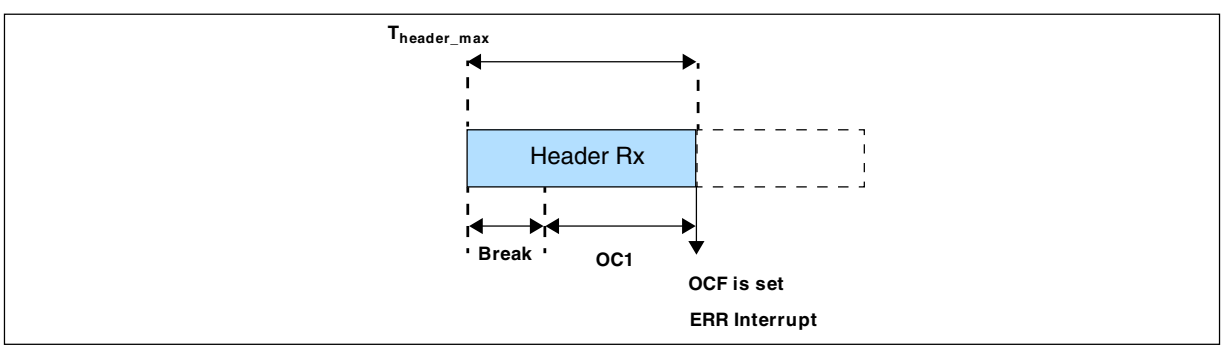


Figure 59-64. Header timeout

59.5.5 UART mode

The following figure shows UART mode configuration.

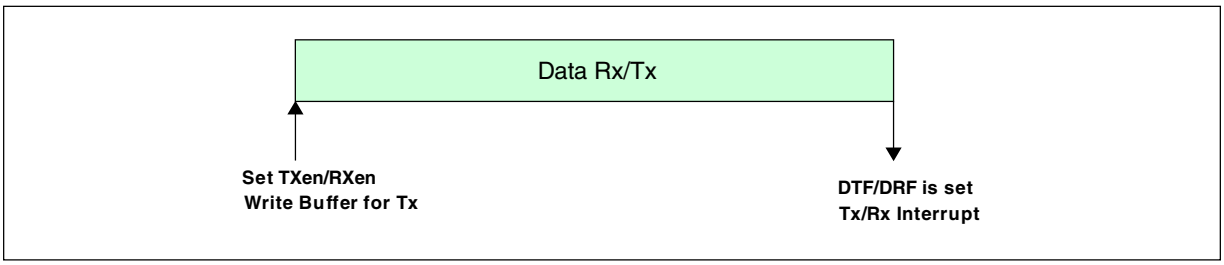


Figure 59-65. UART mode configuration

59.5.6 Interrupts

Interrupt section.

Table 59-19. Interrupts

Interrupt Event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF orSDEF orIDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer Overrun error	BOF	BOIE	Error
UART Timeout error	TO	TOIE	Error
Lin state	Sync Del, Sync Field,Identifier field or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx
Data buffer empty	DBEF	DBEIE	Tx
Data Reception Complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

The following figure shows an Interrupt Flow Diagram.

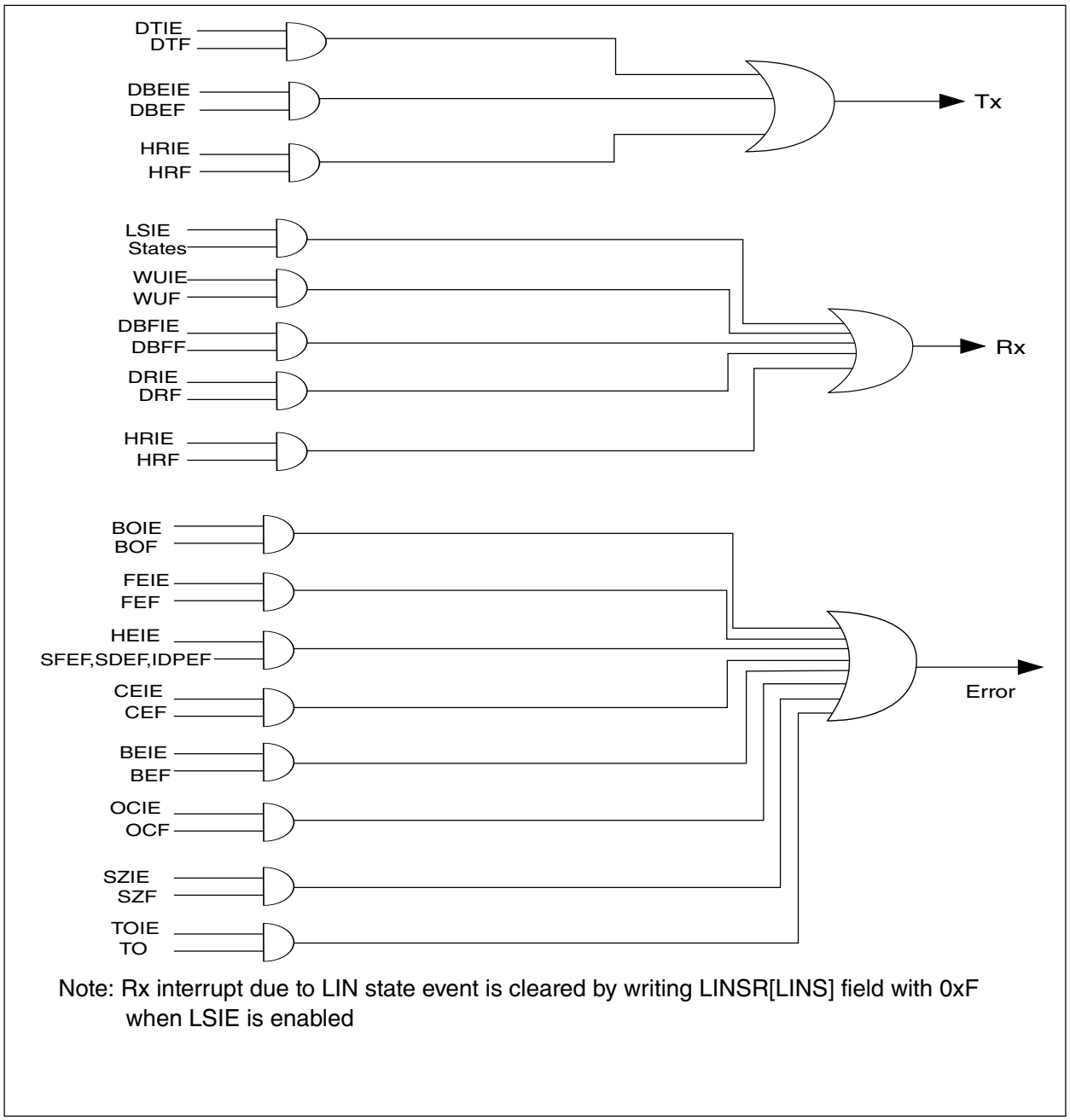


Figure 59-66. Interrupt diagram

59.5.7 LINFlexD Clock Tolerance

1. **Faster receiver tolerance:**
   In this case the receiver has a higher baud rate than the transmitter, thus the stop bit sampling starts already in the last transmitted payload bit. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

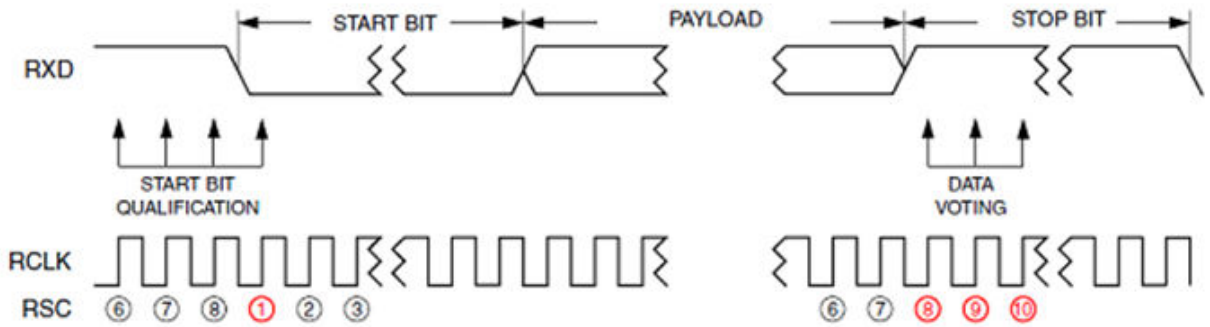


Figure 59-67. Faster receiver

2. **Slower receiver tolerance:**
   In this case the receiver has a slower baud rate than the transmitter, thus the stop bit sampling is still running while the next start bit is already transmitted. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

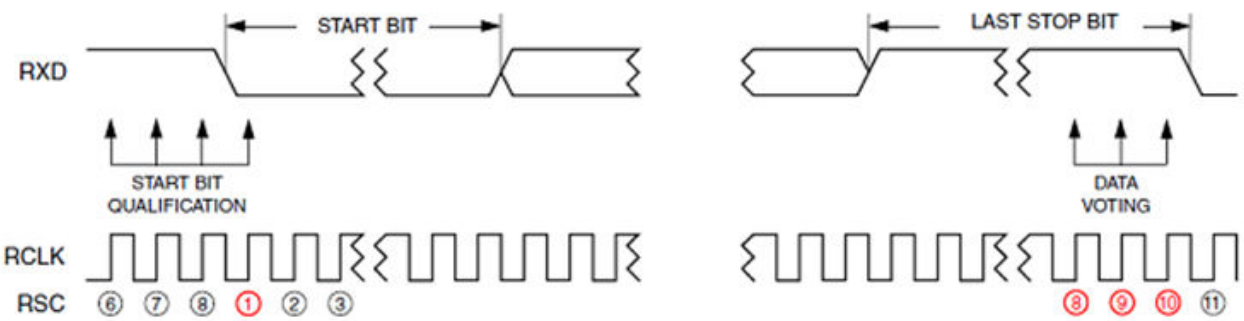


Figure 59-68. Slower receiver

Chapter 60
Reset Generation Module (MC_RGM)

60.1 Introduction

60.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

The following figure shows the MC_RGM block diagram.

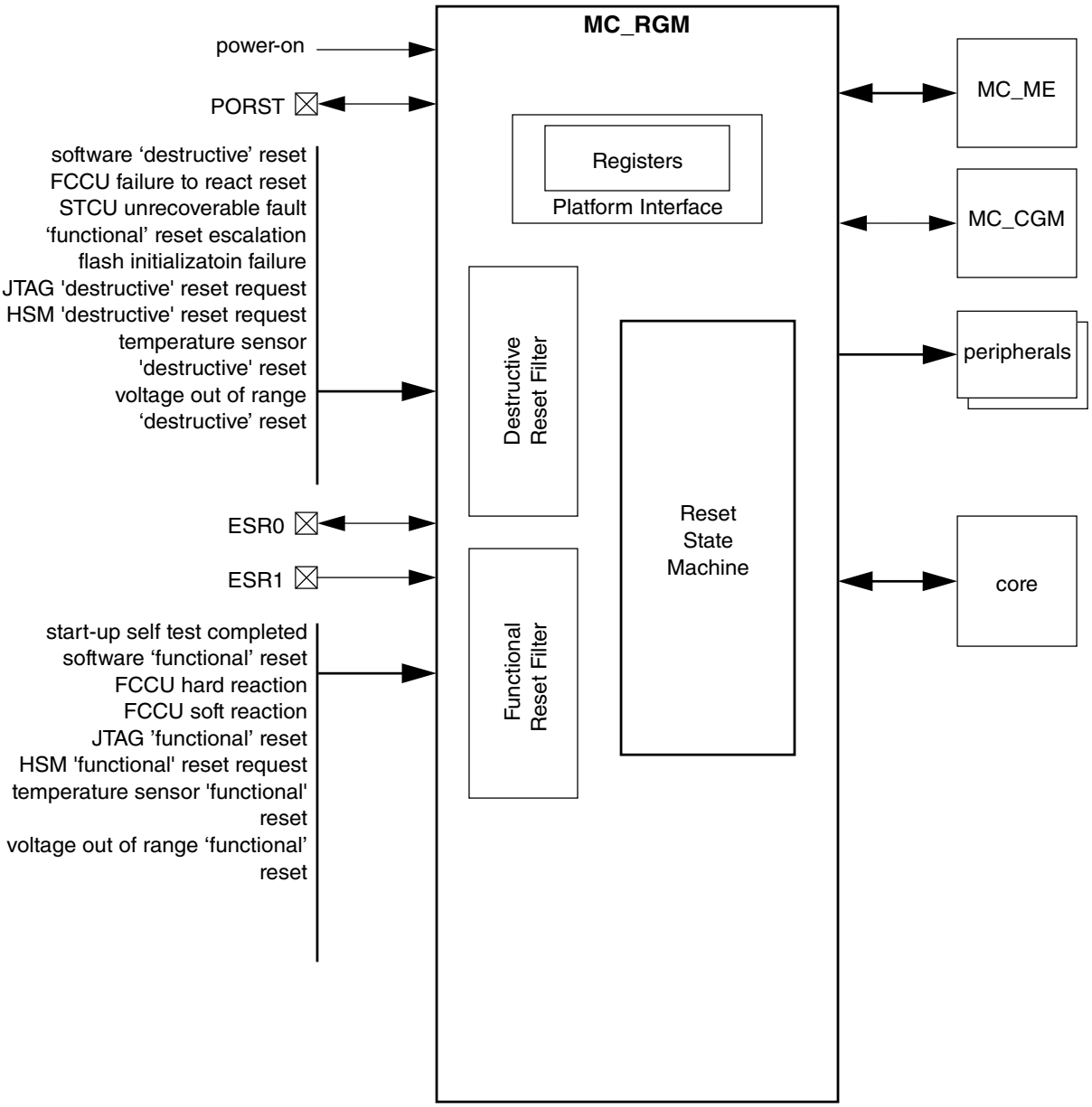


Figure 60-1. MC_RGM Block Diagram

60.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'Destructive' resets management
- 'Functional' resets management
- sSgnalling of reset events after each reset sequence (reset status flags)
- Conversion of reset events to SAFE mode or interrupt request events

- Short reset sequence configuration
- bidirectional reset behavior configuration
- configurable escalation of recurring 'functional' resets to 'destructive' reset
- configurable escalation of recurring 'destructive' resets to keep chip in reset state until next power-on reset

60.1.3 Reset sources

The different reset sources are organized into two families: 'destructive' and 'functional'.

- A 'destructive' reset source is associated with an event related to a critical - usually hardware - error or dysfunction. When a 'destructive' reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a SAFE start-up state for both digital and analog modules, and the memory content must be considered to be unknown except in the software 'destructive' reset and external power-on -reset cases, which do preserve the system memory content if the chip is not configured to execute a selftest on power-on, 'destructive', and external resets. 'Destructive' resets are
  - power-on reset
  - external power-on reset
  - software 'destructive' reset
  - FCCU failure to react reset
  - STCU unrecoverable fault
  - 'functional' reset escalation
  - flash initialization failure
  - JTAG 'destructive' reset request
  - HSM 'destructive' reset request
  - temperature sensor 'destructive' reset
  - voltage out of range 'destructive' reset
- A 'functional' reset source is associated with an event related to a less-critical - usually non-hardware - error or dysfunction. When a 'functional' reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific

digital modules (e.g., debug modules, flash modules) as well as the system memory content is preserved, except on an external reset in the case where the chip is configured to execute a self test on power-on, 'destructive', and external resets. 'Functional' resets are

- ESR0 external reset
- ESR1 external reset
- start-up self test completed
- software 'functional' reset
- FCCU hard reaction
- FCCU soft reaction
- JTAG 'functional' reset
- HSM 'functional' reset request
- temperature sensor 'functional' reset
- voltage out of range 'functional' reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., PHASEn states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (see 'Destructive' Event Reset Disable Register (MC_RGM_DERD) and 'Destructive' Event Alternate Request Register (MC_RGM_DEAR) for 'destructive' resets and 'Functional' Event Reset Disable Register (MC_RGM_FERD) and 'Functional' Event Alternate Request Register (MC_RGM_FEAR) for 'functional' resets).

60.2 External signal description

The MC_RGM interfaces to the bidirectional reset pin ESR0.

60.3 Memory map and register definition

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

NOTE

All registers will be accessible in "User mode" provided the access to this mode is enabled through REGPROT settings. Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error(only if SSCM_ERROR[RAE] is set)

MC_RGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	'Destructive' Event Status Register (MC_RGM_DES)	32	w1c	0000_0001h	60.3.1/3248
10	'Destructive' Event Reset Disable Register (MC_RGM_DERD)	32	R/W	0000_0000h	60.3.2/3250
20	'Destructive' Event Alternate Request Register (MC_RGM_DEAR)	32	R/W	0000_0000h	60.3.3/3253
30	'Destructive' Bidirectional Reset Enable Register (MC_RGM_DBRE)	32	R/W	0000_0000h	60.3.4/3253
300	'Functional' Event Status Register (MC_RGM_FES)	32	w1c	0000_0000h	60.3.5/3256
310	'Functional' Event Reset Disable Register (MC_RGM_FERD)	32	R/W	0000_0000h	60.3.6/3258
320	'Functional' Event Alternate Request Register (MC_RGM_FEAR)	32	R/W	0000_0000h	60.3.7/3260
330	'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)	32	R/W	0180_846Ah	60.3.8/3261
340	'Functional' Event Short Sequence Register (MC_RGM_FESS)	32	R/W	0000_8040h	60.3.9/3263
604	'Functional' Reset Escalation Threshold Register (MC_RGM_FRET)	8	R/W	04h	60.3.10/ 3266
608	'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET)	8	R/W	08h	60.3.11/ 3267

Table continues on the next page...

MC_RGM memory map (continued)

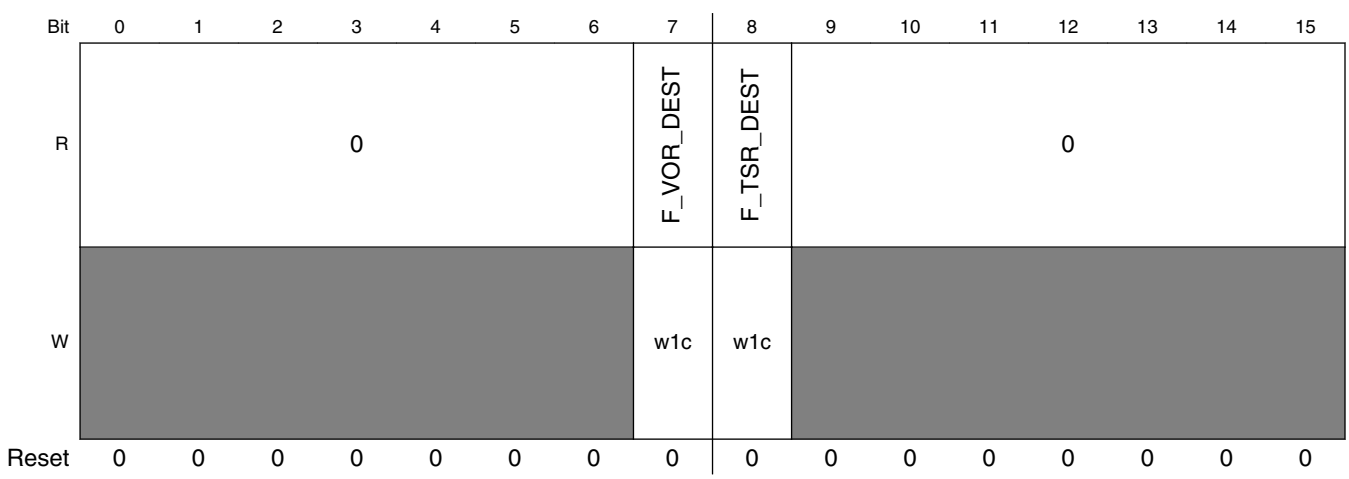
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60C	External Reset Output Extension Control Register (MC_RGM_EROEC)	8	R/W	01h	60.3.12/ 3268
610	Peripheral Reset Register 0 (MC_RGM_PRST0)	32	R/W	0000_0000h	60.3.13/ 3269
614	Peripheral Reset Register 1 (MC_RGM_PRST1)	32	R/W	0000_0000h	60.3.14/ 3270
618	Peripheral Reset Register 2 (MC_RGM_PRST2)	32	R/W	0000_0000h	60.3.15/ 3272
61C	Peripheral Reset Register 3 (MC_RGM_PRST3)	32	R/W	0000_0000h	60.3.16/ 3275
620	Peripheral Reset Register 4 (MC_RGM_PRST4)	32	R/W	0000_0000h	60.3.17/ 3277
624	Peripheral Reset Register 5 (MC_RGM_PRST5)	32	R/W	0000_0000h	60.3.18/ 3278
628	Peripheral Reset Register 6 (MC_RGM_PRST6)	32	R/W	0000_0000h	60.3.19/ 3280
62C	Peripheral Reset Register 7 (MC_RGM_PRST7)	32	R/W	0000_0000h	60.3.20/ 3281

60.3.1 'Destructive' Event Status Register (MC_RGM_DES)

This register contains the status of the 'destructive' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1'.

This register is reset only on power-on.

Address: 0h base + 0h offset = 0h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_DEST	F_SSCM_DEST	0			F_JTAG_DEST	F_FIF	F_EDR	0		F_SUF	F_FFRR	F_SOFT_DEST	0	F_PORST	F_POR
W	w1c	w1c				w1c	w1c	w1c			w1c	w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_RGM_DES field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 F_VOR_DEST	Flag for voltage out of range 'destructive' reset **NOTE:** This bit could become set after POR, but before software can read its value. The state of this bit should be considered indeterminate by software after initial power up 0 No voltage out of range 'destructive' reset event has occurred since the last clear 1 A voltage out of range 'destructive' reset event has occurred
8 F_TSR_DEST	Flag for temperature sensor 'destructive' reset 0 No temperature sensor 'destructive' reset event has occurred since the last clear 1 A temperature sensor 'destructive' reset event has occurred
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 F_HSM_DEST	Flag for HSM 'destructive' reset request 0 No HSM 'destructive' reset request event has occurred since the last clear 1 A HSM 'destructive' reset request event has occurred
17 F_SSCM_DEST	Flag for SSCM 'destructive' reset request 0 No SSCM 'destructive' reset request event has occurred since the last clear 1 An SSCM 'destructive' reset request event has occurred
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 F_JTAG_DEST	Flag for JTAG 'destructive' reset request 0 No JTAG 'destructive' reset request event has occurred since the last clear 1 A JTAG 'destructive' reset request event has occurred
22 F_FIF	Flag for flash initialization failure 0 No flash initialization failure event has occurred since the last clear 1 A flash initialization failure event has occurred

Table continues on the next page...

MC_RGM_DES field descriptions (continued)

Field	Description
23 F_EDR	Flag for 'functional' reset escalation 0 No 'functional' reset escalation event has occurred since the last clear 1 A 'functional' reset escalation event has occurred
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 F_SUF	Flag for STCU unrecoverable fault 0 No STCU unrecoverable fault event has occurred since the last clear 1 A STCU unrecoverable fault event has occurred
27 F_FFRR	Flag for FCCU failure to react reset 0 No FCCU failure to react reset event has occurred since the last clear 1 A FCCU failure to react reset event has occurred
28 F_SOFT_DEST	Flag for software 'destructive' reset 0 No software 'destructive' reset event has occurred since the last clear 1 A software 'destructive' reset event has occurred
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 F_PORST	Flag for external power-on reset 0 No external power-on reset has occurred since the last clear 1 A external power-on reset event has occurred
31 F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

60.3.2 'Destructive' Event Reset Disable Register (MC_RGM_DERD)

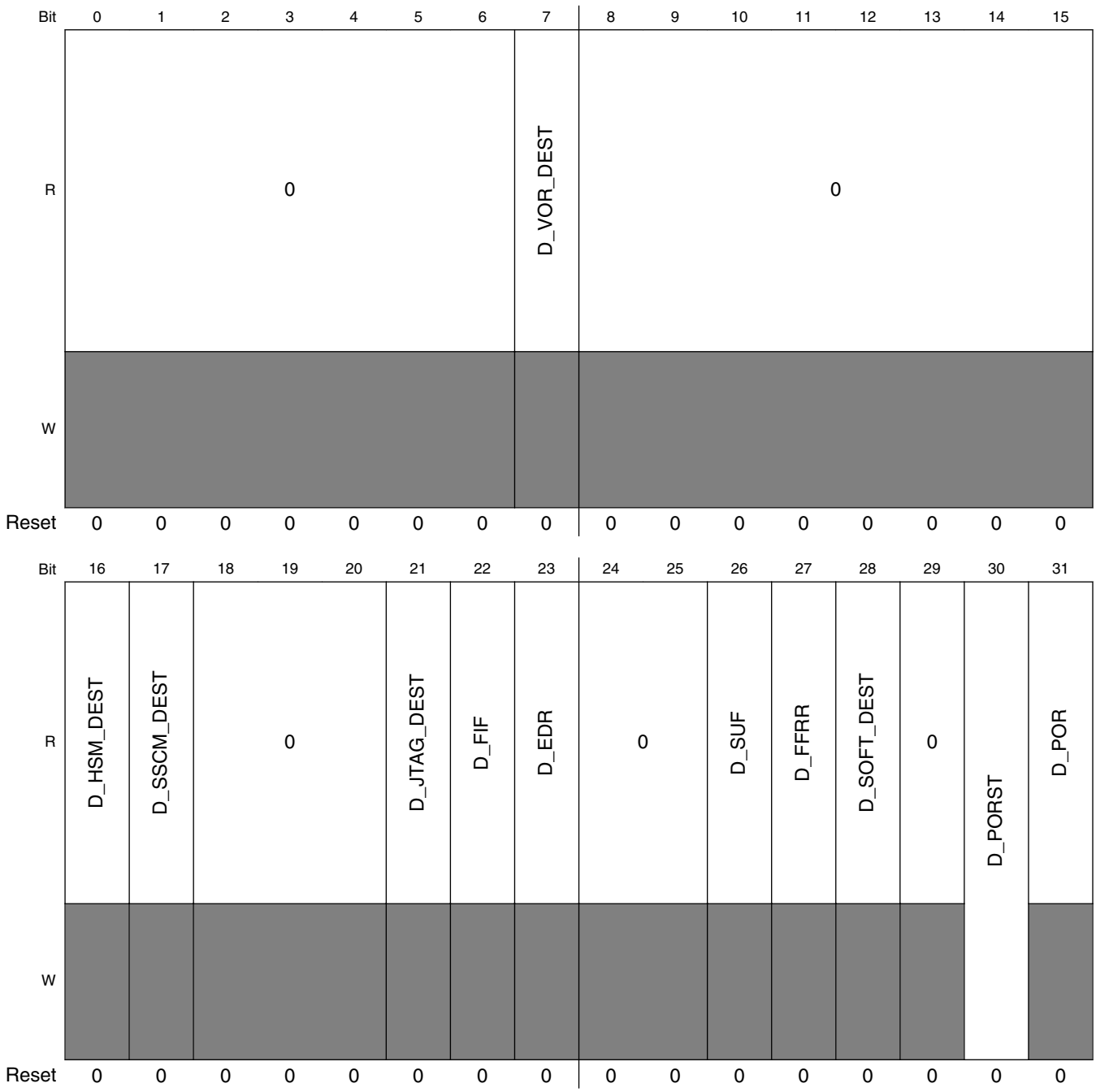
This register provides dedicated bits to disable particular 'destructive' reset sources. When a 'destructive' reset source is disabled, the associated 'destructive' event will trigger either a SAFE mode request or an interrupt request (see 'Destructive' Event Alternate Request Register (MC_RGM_DEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on.

Caution

It is important to clear the RGM_DES register before setting any of the bits in the RGM_DERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0h base + 10h offset = 10h



MC_RGM_DERD field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 D_VOR_DEST	Disable voltage out of range 'destructive' reset 0 A voltage out of range 'destructive' reset event triggers a reset sequence

Table continues on the next page...

MC_RGM_DERD field descriptions (continued)

Field	Description
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 D_HSM_DEST	Disable HSM 'destructive' reset request 0 A HSM 'destructive' reset request event triggers a reset sequence
17 D_SSCM_DEST	Disable SSCM 'destructive' reset request 0 An SSCM 'destructive' reset request event triggers a reset sequence
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_JTAG_DEST	Disable JTAG 'destructive' reset request 0 A JTAG 'destructive' reset request event triggers a reset sequence
22 D_FIF	Disable flash initialization failure 0 A flash initialization failure event triggers a reset sequence
23 D_EDR	Disable 'functional' reset escalation 0 A 'functional' reset escalation event triggers a reset sequence
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 D_SUF	Disable STCU unrecoverable fault 0 A STCU unrecoverable fault event triggers a reset sequence
27 D_FFRR	Disable FCCU failure to react reset 0 A FCCU failure to react reset event triggers a reset sequence
28 D_SOFT_DEST	Disable software 'destructive' reset 0 A software 'destructive' reset triggers a reset sequence
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 D_PORST	Disable external power-on reset 0 An external power-on reset event triggers a reset sequence 1 An external power-on reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR
31 D_POR	Disable Power-On reset 0 A power-on event triggers a reset sequence

60.3.3 'Destructive' Event Alternate Request Register (MC_RGM_DEAR)

This register defines an alternate request to be generated when a reset on a 'destructive' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														AR_PORST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_DEAR field descriptions

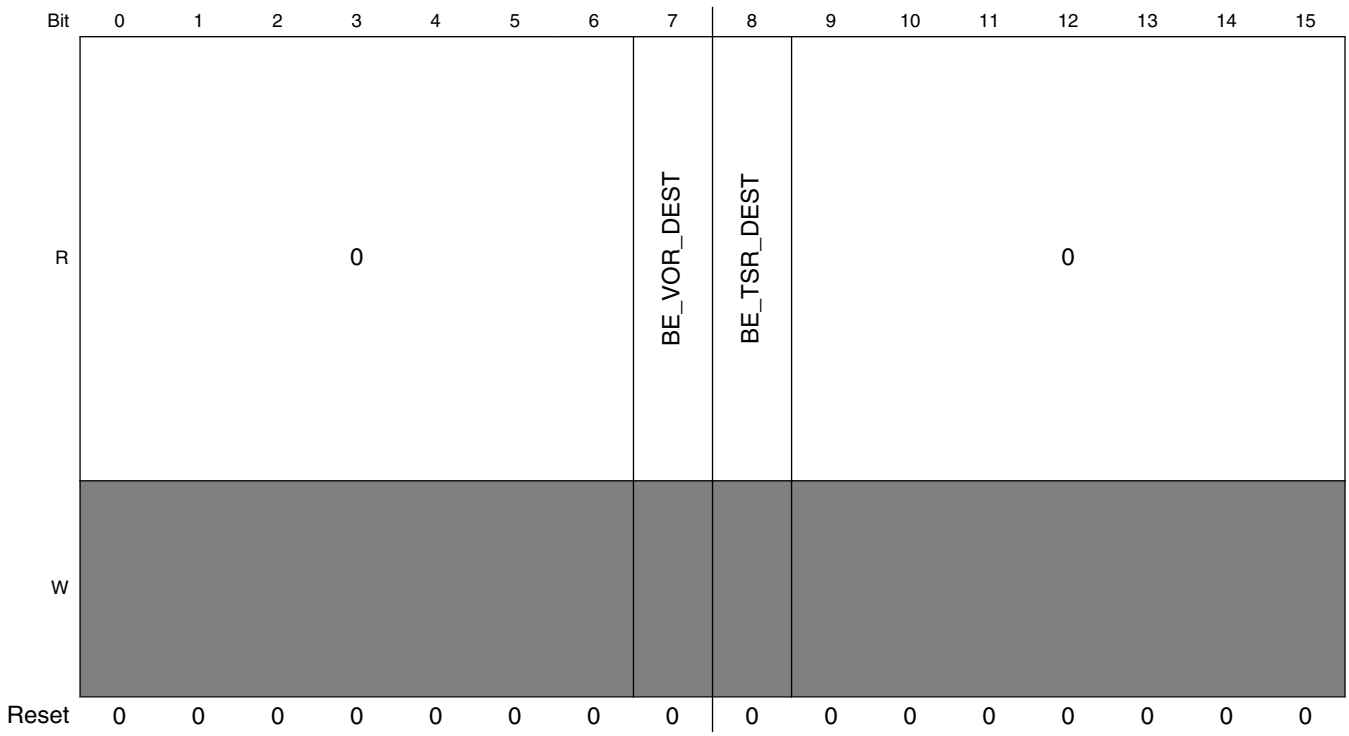
Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 AR_PORST	Alternate Request for external power-on reset 0 Generate a SAFE mode request on an external power-on reset event if the reset is disabled 1 Generate an interrupt request on an external power-on reset event if the reset is disabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.3.4 'Destructive' Bidirectional Reset Enable Register (MC_RGM_DBRE)

This register enables the generation of an external reset on 'destructive' reset. It can be accessed as read only in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0h base + 30h offset = 30h



MC_RGM_DBRE field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_DBRE field descriptions (continued)

Field	Description
7 BE_VOR_DEST	Bidirectional Reset Enable for voltage out of range 'destructive' reset 0 ESR0 is asserted on a voltage out of range 'destructive' reset event
8 BE_TSR_DEST	Bidirectional Reset Enable for temperature sensor 'destructive' reset 0 ESR0 is asserted on a temperature sensor 'destructive' reset event if the reset is enabled
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 BE_HSM_DEST	Bidirectional Reset Enable for HSM 'destructive' reset request 0 ESR0 is asserted on a HSM 'destructive' reset request event
17 BE_SSCM_ DEST	Bidirectional Reset Enable for SSCM 'destructive' reset request 0 ESR0 is asserted on a SSCM 'destructive' reset request event
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BE_JTAG_DEST	Bidirectional Reset Enable for JTAG 'destructive' reset request 0 ESR0 is asserted on a JTAG 'destructive' reset request event
22 BE_FIF	Bidirectional Reset Enable for flash initialization failure 0 ESR0 is asserted on a flash initialization failure event
23 BE_EDR	Bidirectional Reset Enable for 'functional' reset escalation 0 ESR0 is asserted on a 'functional' reset escalation event
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 BE_SUF	Bidirectional Reset Enable for STCU unrecoverable fault 0 ESR0 is asserted on a STCU unrecoverable fault event
27 BE_FFRR	Bidirectional Reset Enable for FCCU failure to react reset 0 ESR0 is asserted on a FCCU failure to react reset event
28 BE_SOFT_DEST	Bidirectional Reset Enable for software 'destructive' reset 0 ESR0 is asserted on a software 'destructive' reset event
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 BE_PORST	Bidirectional Reset Enable for external power-on reset 0 ESR0 is asserted on an external power-on reset event if the reset is enabled
31 BE_POR	Bidirectional Reset Enable for Power-On reset 0 ESR0 is asserted on a Power-On reset event

60.3.5 'Functional' Event Status Register (MC_RGM_FES)

This register contains the status of the 'functional' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1' if the triggering event has already been cleared at the source.

This register is reset only on power-on.

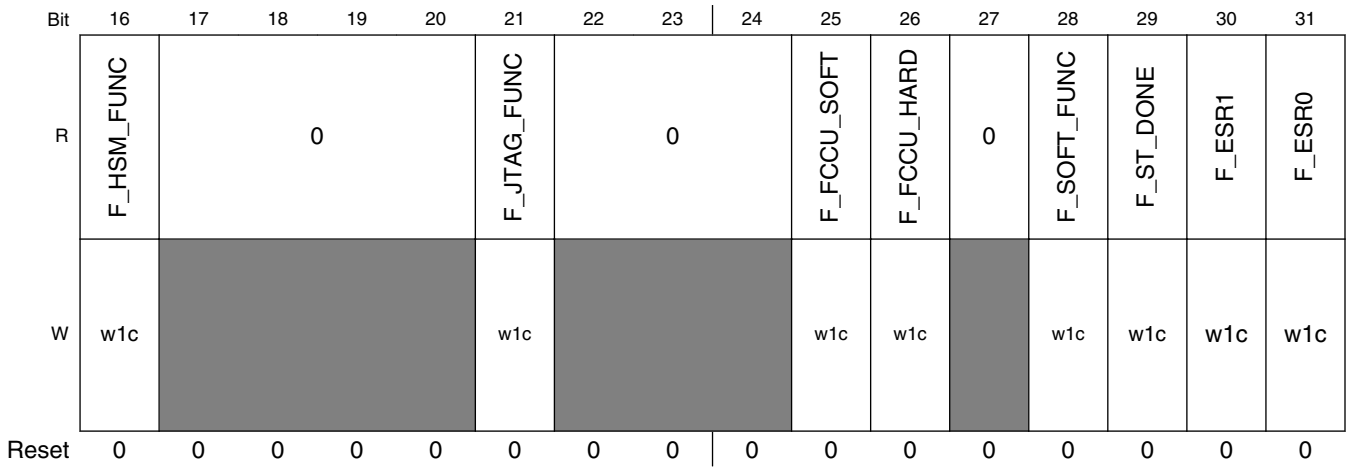
NOTE

If a 'functional' reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES has been properly cleared, and if not, clear it again.

When ESR0 gate is disabled via UTEST_Miscellaneous[ESR0_Gate] field of DCF client and ESR0 is asserted by setting the SIUL_SCR0[ESR0_ASSERT] field, and then PORST is asserted or a destructive reset occurs and the system comes up, the RGM_FES register reads a value of 0x00000001 indicating that an ESR0 reset has also occurred.

Address: 0h base + 300h offset = 300h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							F_VOR_FUNC	F_TSR_FUNC	0						
W								w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MC_RGM_FES field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 F_VOR_FUNC	Flag for voltage out of range 'functional' reset 0 No voltage out of range 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range 'functional' reset event has occurred
8 F_TSR_FUNC	Flag for temperature sensor 'functional' reset 0 No temperature sensor 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A temperature sensor 'functional' reset event has occurred
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 F_HSM_FUNC	Flag for HSM 'functional' reset request 0 No HSM 'functional' reset request event has occurred since either the last clear or the last power-on reset assertion 1 A HSM 'functional' reset request event has occurred
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 F_JTAG_FUNC	Flag for JTAG 'functional' reset 0 No JTAG 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A JTAG 'functional' reset event has occurred
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 F_FCCU_SOFT	Flag for FCCU soft reaction 0 No FCCU soft reaction event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU soft reaction event has occurred

Table continues on the next page...

MC_RGM_FES field descriptions (continued)

Field	Description
26 F_FCCU_HARD	Flag for FCCU hard reaction reset 0 No FCCU hard reaction reset event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU hard reaction reset event has occurred
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 F_SOFT_FUNC	Flag for software 'functional' reset 0 No software 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A software 'functional' reset event has occurred
29 F_ST_DONE	Flag for self test completed 0 No self test completed event has occurred since either the last clear or the last power-on reset assertion 1 A self test completed event has occurred
30 F_ESR1	Flag for ESR1 External Reset 0 No ESR1 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR1 external reset event has occurred
31 F_ESR0	Flag for ESR0 External Reset 0 No ESR0 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR0 external reset event has occurred

60.3.6 'Functional' Event Reset Disable Register (MC_RGM_FERD)

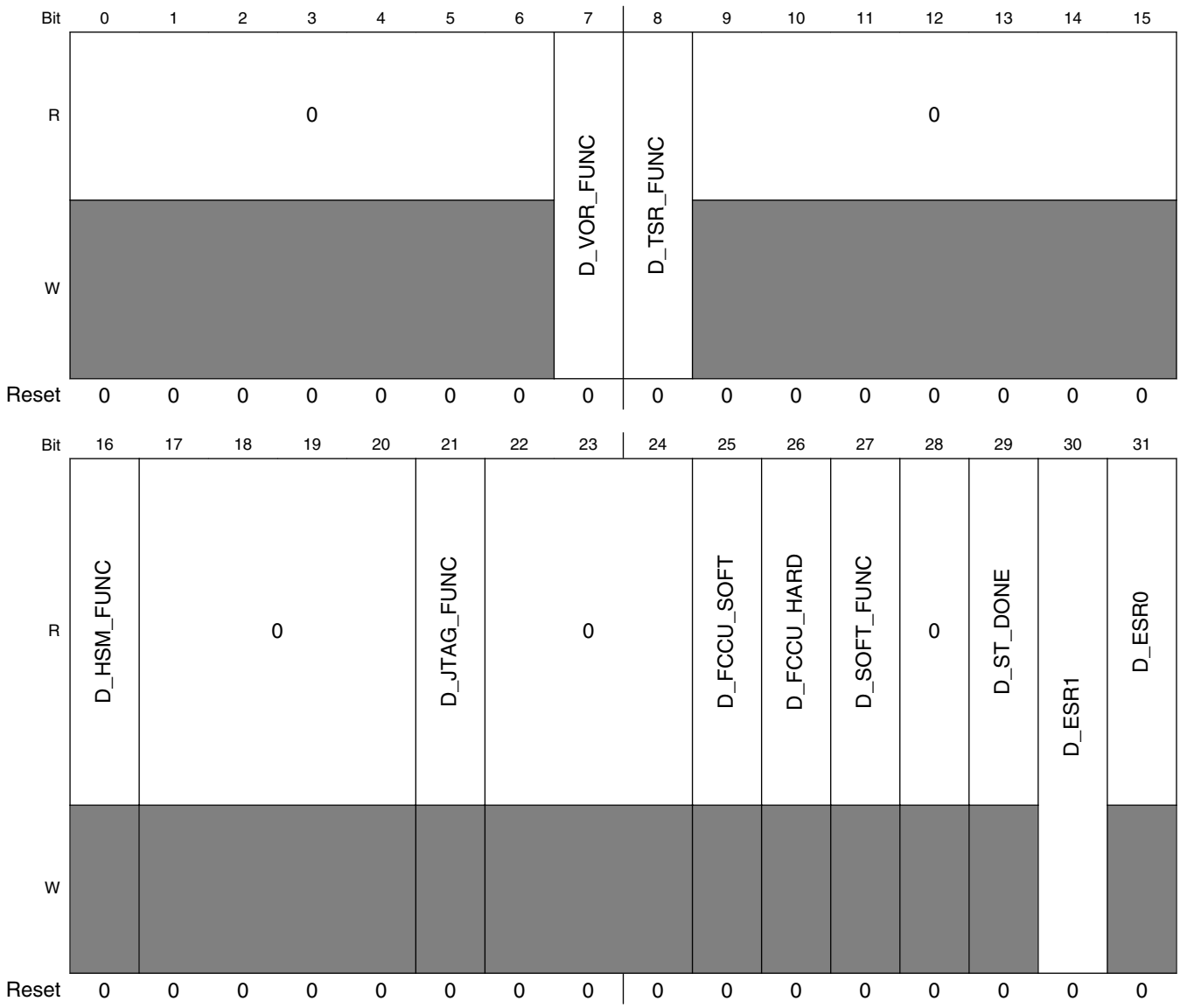
This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see 'Functional' Event Alternate Request Register (MC_RGM_FEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

This register is reset only on power-on and any 'destructive' reset.

Caution

It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0h base + 310h offset = 310h



MC_RGM_FERD field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 D_VOR_FUNC	Disable voltage out of range 'functional' reset 0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
8 D_TSR_FUNC	Disable temperature sensor 'functional' reset 0 A temperature sensor 'functional' reset event triggers a reset sequence 1 A temperature sensor 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_TSR_FUNC

Table continues on the next page...

MC_RGM_FERD field descriptions (continued)

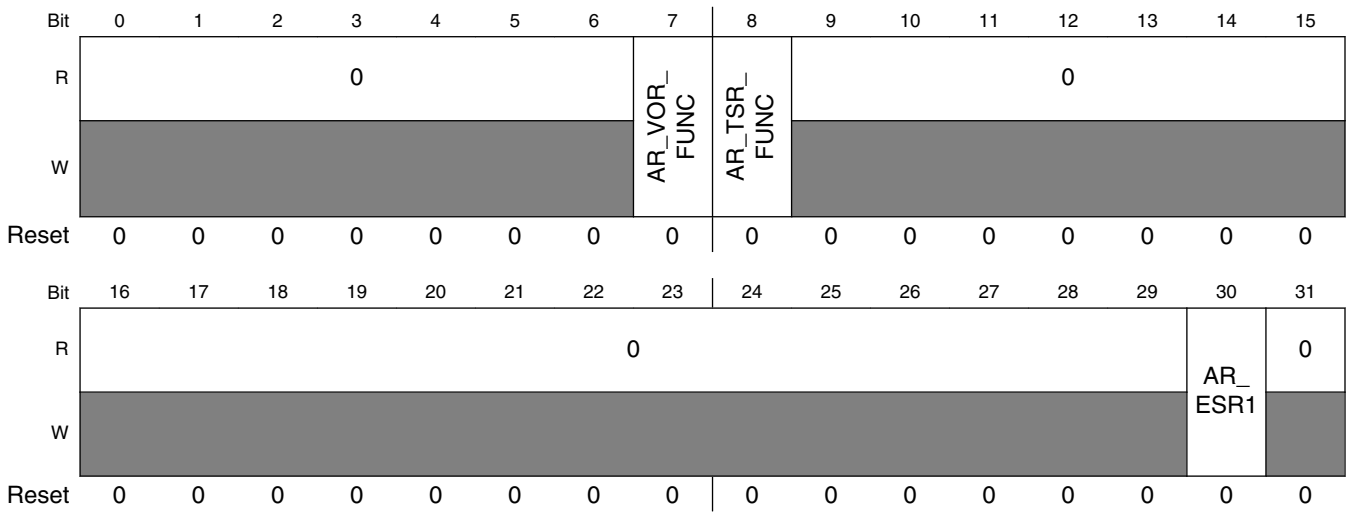
Field	Description
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 D_HSM_FUNC	Disable HSM 'functional' reset request 0 A HSM 'functional' reset request event triggers a reset sequence
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_JTAG_FUNC	Disable JTAG 'functional' reset 0 A JTAG 'functional' reset event triggers a reset sequence
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 D_FCCU_SOFT	Disable FCCU soft reaction 0 A FCCU soft reaction event triggers a reset sequence
26 D_FCCU_HARD	Disable FCCU hard reaction reset 0 A FCCU hard reaction reset event triggers a reset sequence
27 D_SOFT_FUNC	software 'functional' reset 0 A software 'functional' reset triggers a reset sequence.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 D_ST_DONE	Disable self test completed 0 A self test completed event triggers a reset sequence
30 D_ESR1	Disable ESR1 External Reset 0 An ESR1 external reset event triggers a reset sequence 1 An ESR1 external reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_ESR1
31 D_ESR0	Disable ESR0 External Reset 0 An ESR0 external reset event triggers a reset sequence

60.3.7 'Functional' Event Alternate Request Register (MC_RGM_FEAR)

This register defines an alternate request to be generated when a reset on a 'functional' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 320h offset = 320h



MC_RGM_FEAR field descriptions

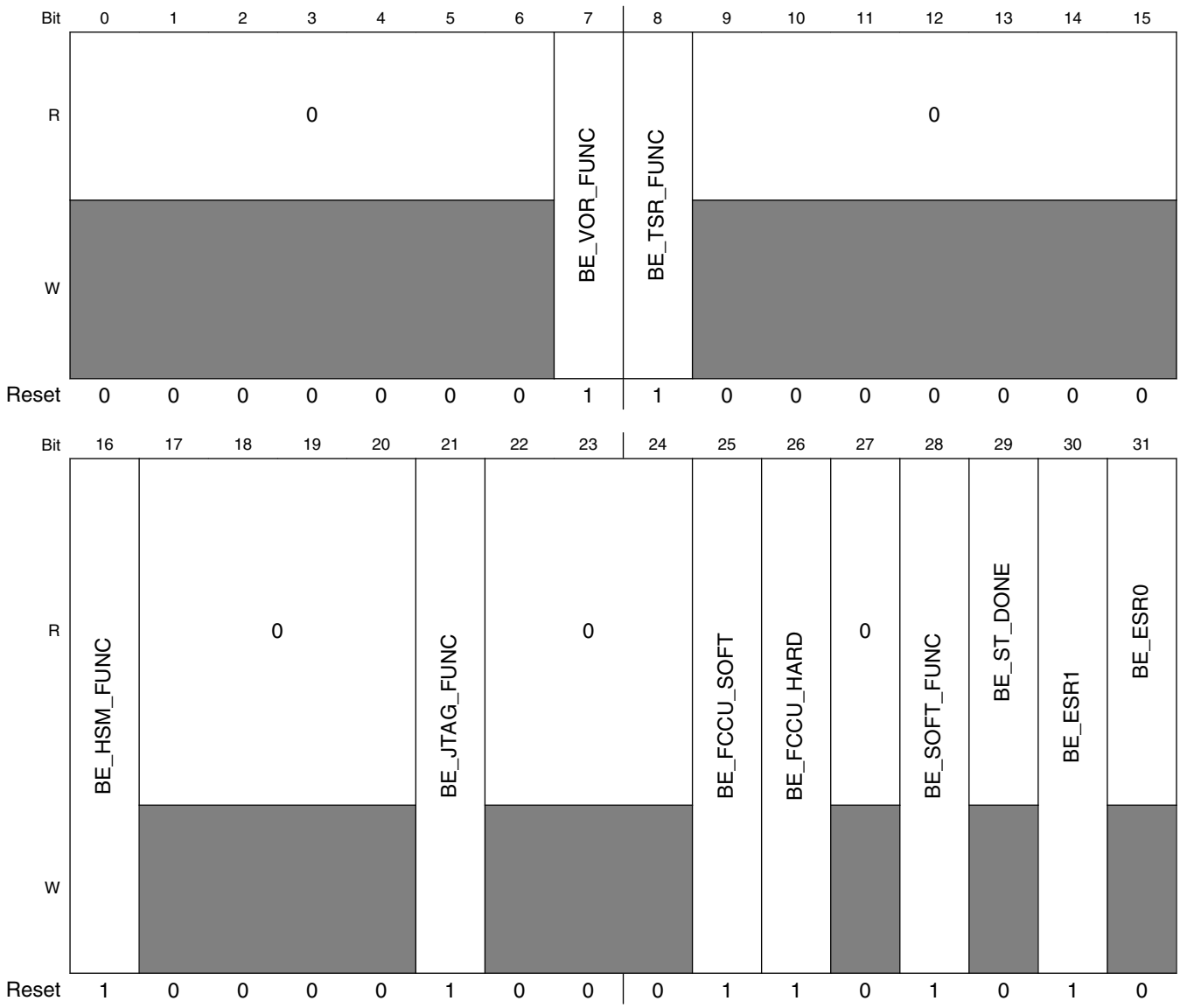
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 AR_VOR_FUNC	Alternate Request for voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
8 AR_TSR_FUNC	Alternate Request for temperature sensor 'functional' reset 0 Generate a SAFE mode request on a temperature sensor 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'functional' reset event if the reset is disabled
9–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 AR_ESR1	Alternate Request for ESR1 External Reset 0 Generate a SAFE mode request on an ESR1 external reset event if the reset is disabled 1 Generate an interrupt request on an ESR1 external reset event if the reset is disabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.3.8 'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 330h offset = 330h



MC_RGM_FBRE field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 BE_VOR_FUNC	Bidirectional Reset Enable for voltage out of range 'functional' reset 0 ESR0 is asserted on a voltage out of range 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a voltage out of range 'functional' reset event
8 BE_TSR_FUNC	Bidirectional Reset Enable for temperature sensor 'functional' reset 0 ESR0 is asserted on a temperature sensor 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a temperature sensor 'functional' reset event
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_FBRE field descriptions (continued)

Field	Description
16 BE_HSM_FUNC	Bidirectional Reset Enable for HSM 'functional' reset request 0 ESR0 is asserted on a HSM 'functional' reset request event 1 ESR0 is not asserted on a HSM 'functional' reset request event
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG 'functional' reset 0 ESR0 is asserted on a JTAG 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a JTAG 'functional' reset event
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 BE_FCCU_SOFT	Bidirectional Reset Enable for FCCU soft reaction 0 ESR0 is asserted on a FCCU soft reaction event 1 ESR0 is not asserted on a FCCU soft reaction event
26 BE_FCCU_ HARD	Bidirectional Reset Enable for a FCCU hard reaction 0 ESR0 is asserted on a FCCU hard reaction reset event 1 ESR0 is not asserted on a FCCU hard reaction reset event
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 BE_SOFT_FUNC	Bidirectional Reset Enable for software 'functional' reset 0 ESR0 is asserted on a software 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a software 'functional' reset event
29 BE_ST_DONE	Bidirectional Reset Enable for self test completed 0 ESR0 is asserted on a self test completed event
30 BE_ESR1	Bidirectional Reset Enable for ESR1 External Reset 0 ESR0 is asserted on an ESR1 external reset event if the reset is enabled 1 ESR0 is not asserted on an ESR1 external reset event
31 BE_ESR0	Bidirectional Reset Enable for ESR0 External Reset 0 ESR0 is asserted on an ESR0 external reset event if the reset is enabled

60.3.9 'Functional' Event Short Sequence Register (MC_RGM_FESS)

This register defines which reset sequence will be done when a 'functional' reset sequence is triggered. The 'functional' reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

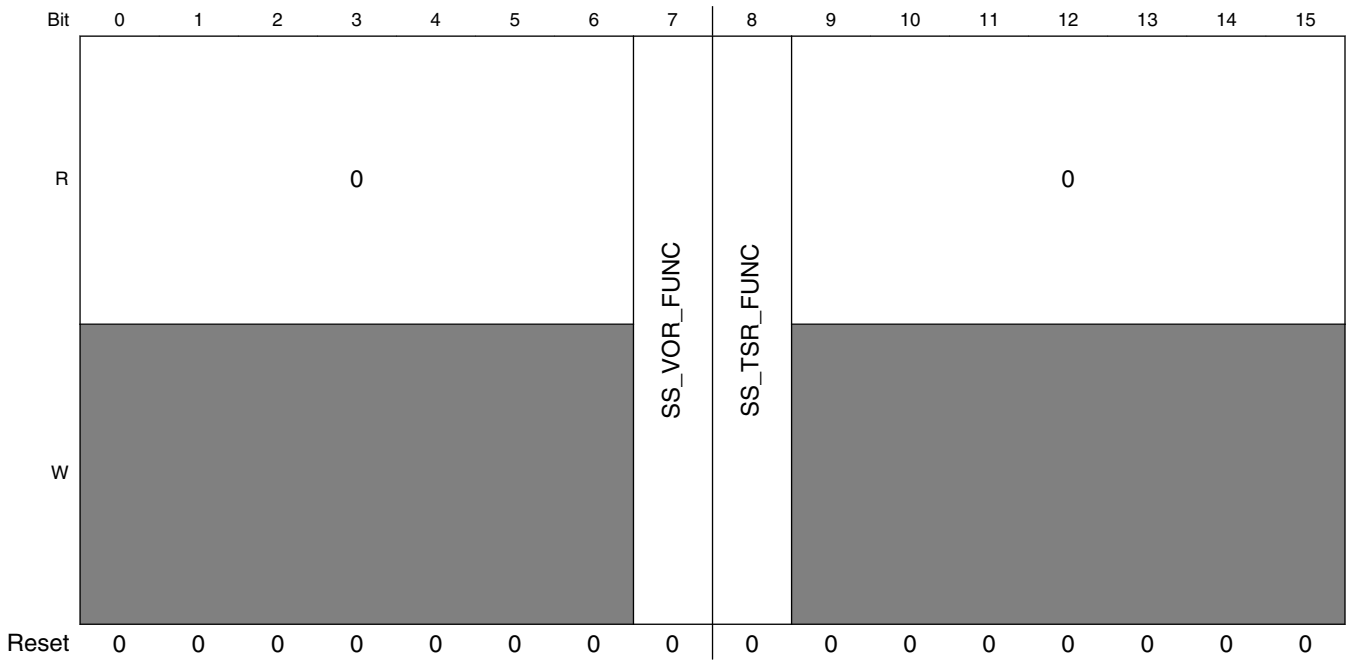
NOTE

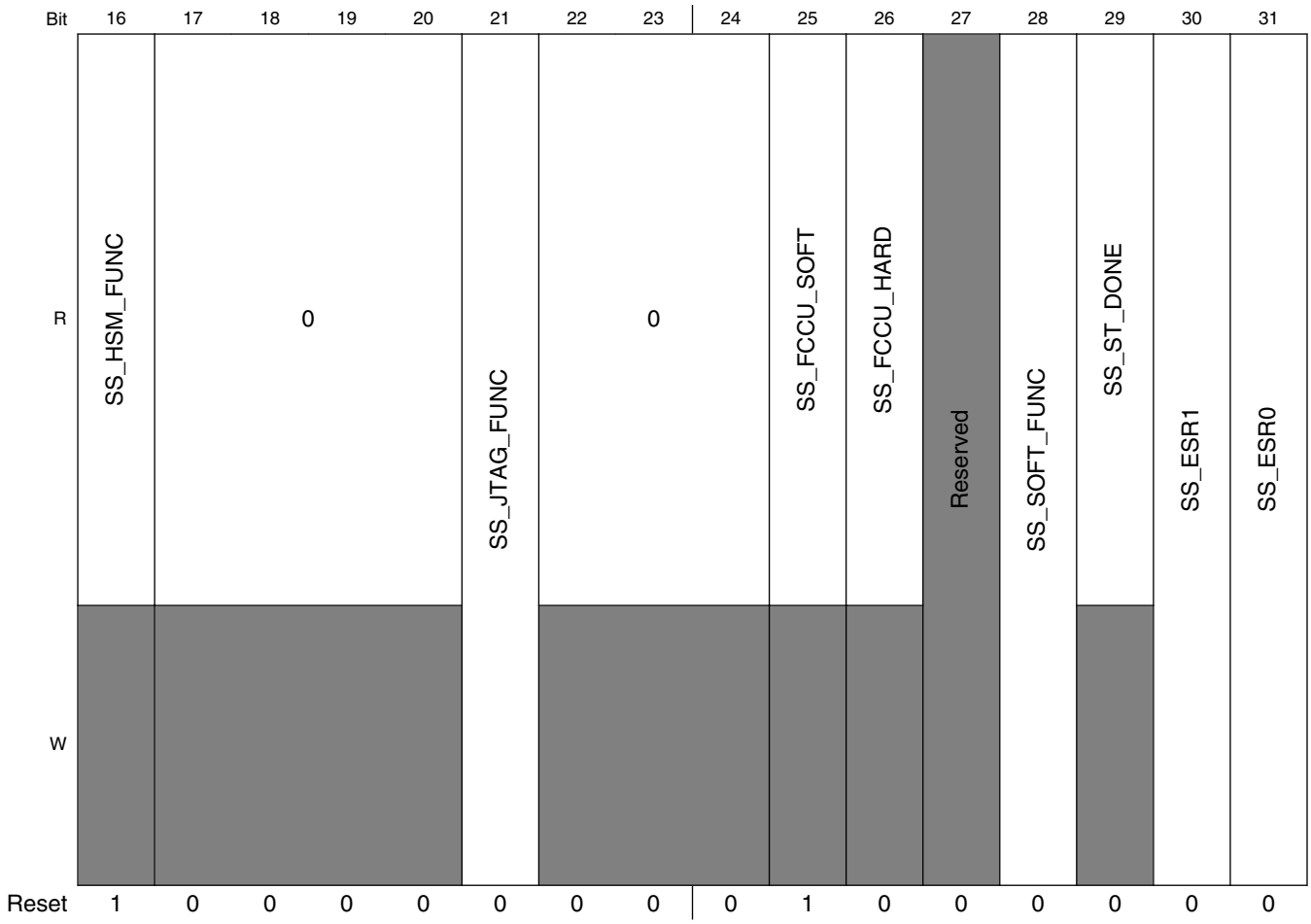
This could be useful for fast reset sequence, for example to skip flash reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 340h offset = 340h





MC_RGM_FESS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SS_VOR_FUNC	Short Sequence for voltage out of range 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
8 SS_TSR_FUNC	Short Sequence for temperature sensor 'functional' reset 0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SS_HSM_FUNC	Short Sequence for HSM 'functional' reset request 1 The reset sequence triggered by a HSM 'functional' reset request event will start from PHASE3, skipping PHASE1 and PHASE2

Table continues on the next page...

MC_RGM_FESS field descriptions (continued)

Field	Description
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SS_JTAG_FUNC	Short Sequence for JTAG 'functional' reset 0 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SS_FCCU_SOFT	Short Sequence for FCCU soft reaction 1 The reset sequence triggered by a FCCU soft reaction event will start from PHASE3, skipping PHASE1 and PHASE2
26 SS_FCCU_ HARD	Short Sequence for a FCCU hard reaction 0 The reset sequence triggered by a FCCU hard reaction reset event will start from PHASE1
27 Reserved	This field is reserved.
28 SS_SOFT_FUNC	Short Sequence for software 'functional' reset 0 The reset sequence triggered by a software 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a software 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
29 SS_ST_DONE	Short Sequence for self test completed 0 The reset sequence triggered by a self test completed event will start from PHASE1
30 SS_ESR1	Short Sequence for ESR1 External Reset 0 The reset sequence triggered by an ESR1 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR1 external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2
31 SS_ESR0	Short Sequence for ESR0 External Reset 0 The reset sequence triggered by an ESR0 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR0 external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2

60.3.10 'Functional' Reset Escalation Threshold Register (MC_RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the FRET field will enable the 'functional' reset escalation function. Writing any value to this register will reset the 'functional' reset counter. See 'Functional' reset escalation for details on the 'functional' reset escalation function.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7
Read	0				FRET			
Write								
Reset	0	0	0	0	0	1	0	0

MC_RGM_FRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 FRET	'Functional' Reset Escalation Threshold - If the value of this field is 0, the 'functional' reset escalation function is disabled. Any other value is the number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register isn't written to beforehand.

60.3.11 'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET)

This register sets the threshold for 'destructive' reset escalation topti_top.v keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the DRET field will enable the 'destructive' reset software escalation function. Writing any value to this register will reset the 'destructive' reset counter. See 'Destructive' reset escalation for details on the 'destructive' reset escalation function.

This register is reset only on power-on.

This register is reset only on power-on.

NOTE

The destructive resets which can trigger escalation are F_SOFT_DEST, F_FFRR, F_SUF, F_EDR and F_FIF. It is defined by the MC_RGM parameter DEST_ESC_INCR_TRIG.

Address: 0h base + 608h offset = 608h

Bit	0	1	2	3	4	5	6	7
Read	0				DRET			
Write								
Reset	0	0	0	0	1	0	0	0

MC_RGM_DRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 DRET	'Destructive' Reset Escalation Threshold - If the value of this field is 0, the 'destructive' reset escalation function is disabled. Any other value is the number of 'destructive' resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register isn't written to beforehand.

60.3.12 External Reset Output Extension Control Register (MC_RGM_EROEC)

This register controls the assertion of the ESR0 pin after the reset sequence has completed. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. The register bit is cleared on write '1'.

Clearing the EROEC bit will cause the MC_RGM to stop asserting the ESR0 pin within two 16 MHz internal RC oscillator clock cycles. The ESR0 pin cannot be asserted by writing to this register.

This register also provides the current value sampled on ESR0 via the ERIS bit. This is useful for software to poll if, for example, ESR0 is used in the system to synchronize the application software with an external event.

This register is reset only on power-on.

Address: 0h base + 60Ch offset = 60Ch

Bit	0	1	2	3	4	5	6	7
Read	0						ERIS	EROEC
Write								w1c
Reset	0	0	0	0	0	0	0	1

MC_RGM_EROEC field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ERIS	External Reset Input Status

Table continues on the next page...

MC_RGM_EROEC field descriptions (continued)

Field	Description
	0 A '0' is being observed on ESR0 1 A '1' is being observed on ESR0
7 EROEC	External Reset Output Assertion Control 0 ESR0 is not being asserted by the MC_RGM 1 ESR0 is being asserted by the MC_RGM

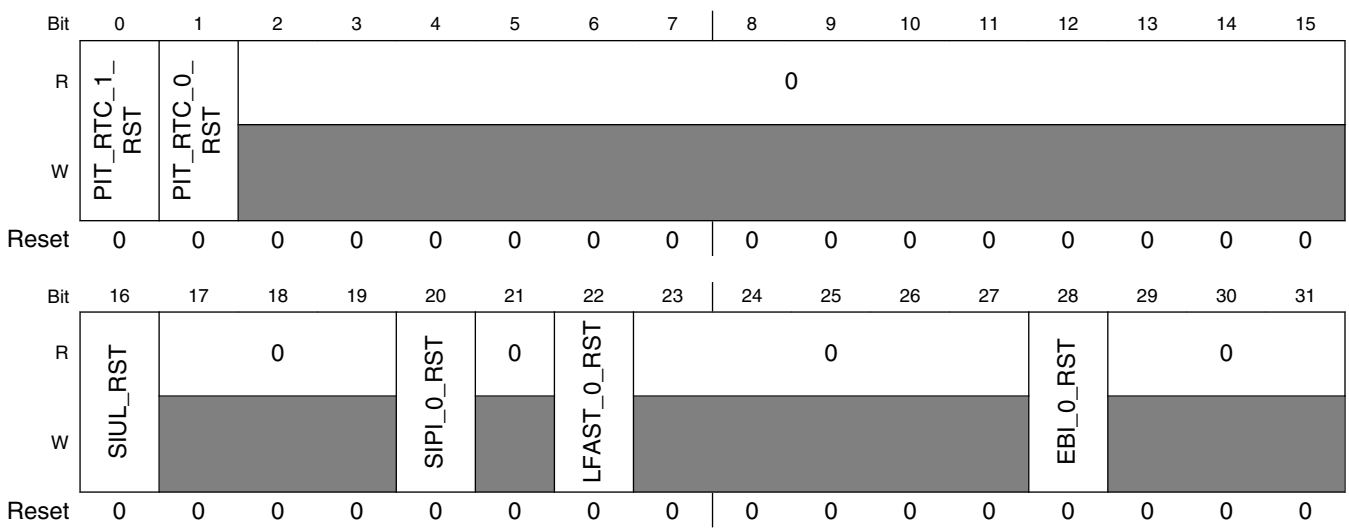
60.3.13 Peripheral Reset Register 0 (MC_RGM_PRST0)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 610h offset = 610h



MC_RGM_PRST0 field descriptions

Field	Description
0 PIT_RTC_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

Table continues on the next page...

MC_RGM_PRST0 field descriptions (continued)

Field	Description
1 PIT_RTC_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
2–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SIUL_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 SIPI_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LFAST_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
23–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 EBI_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

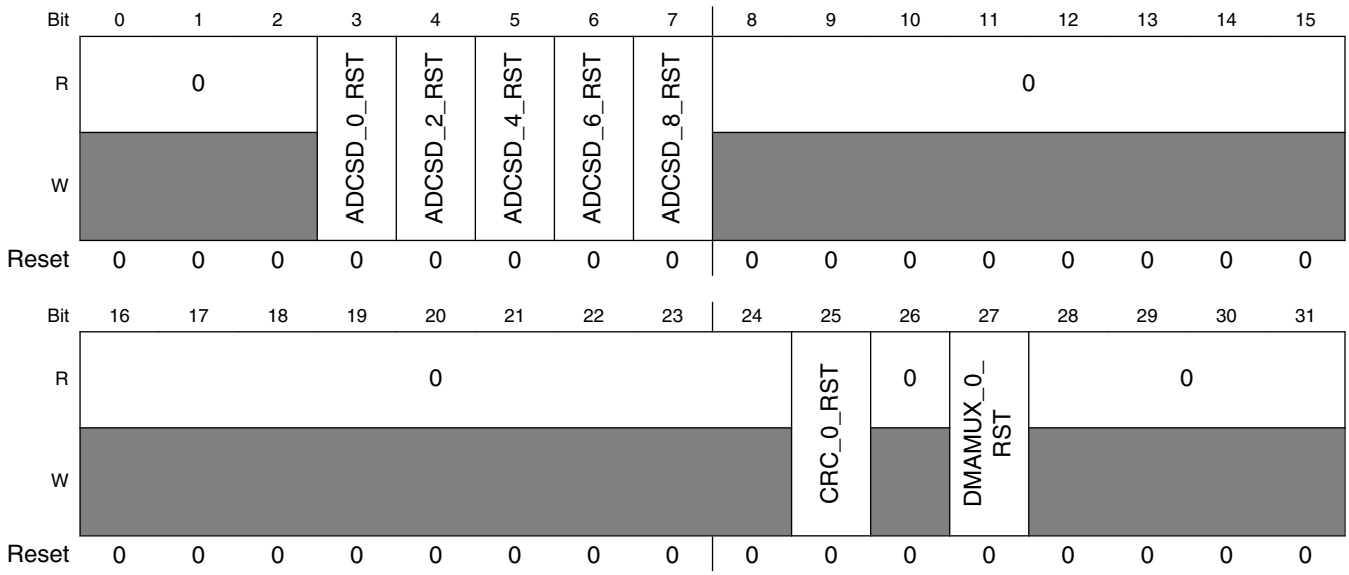
60.3.14 Peripheral Reset Register 1 (MC_RGM_PRST1)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 614h offset = 614h



MC_RGM_PRST1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ADCSD_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
4 ADCSD_2_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5 ADCSD_4_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
6 ADCSD_6_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect.

Table continues on the next page...

MC_RGM_PRST1 field descriptions (continued)

Field	Description
	0 no forced reset of peripheral 1 forced reset of peripheral
7 ADCSD_8_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
8–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 CRC_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 DMAMUX_0_ RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.3.15 Peripheral Reset Register 2 (MC_RGM_PRST2)

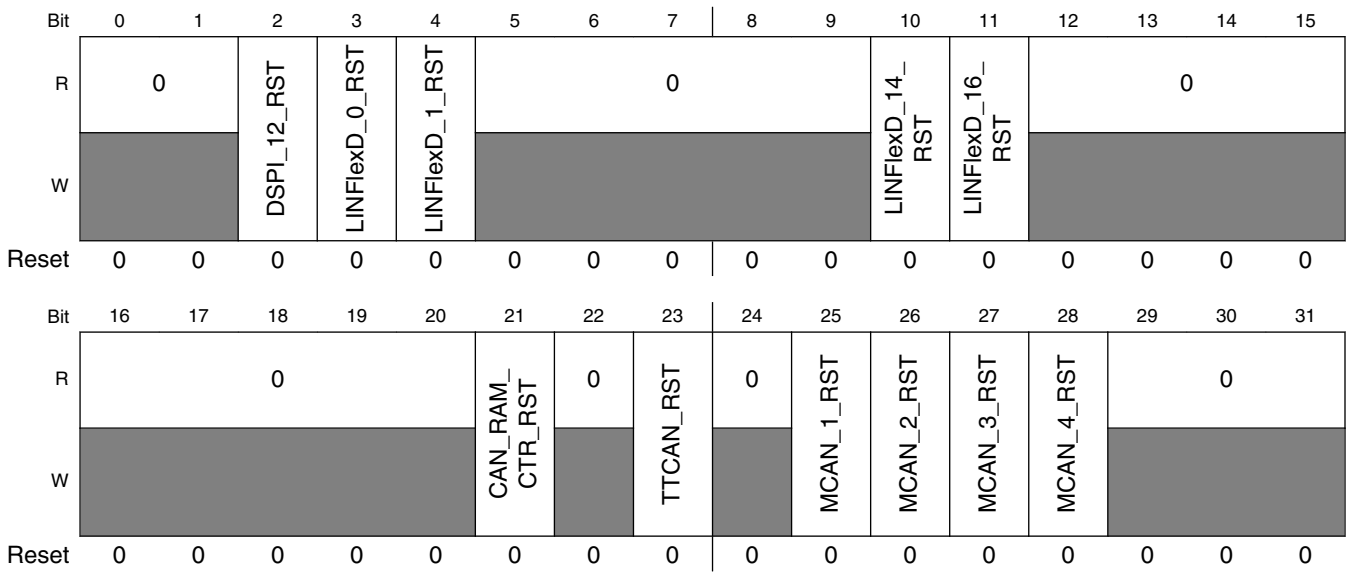
This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

In addition to resetting the CAN_RAM_CTR module, writing a '1' to the CAN_RAM_CTR_RST bit will also reset the MCAN_1, MCAN_2, and MCAN_3 modules and keep them in reset until a '0' is written to this bit. MCAN_1, MCAN_2, and MCAN_3 can still be reset individually via their individual bits.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 618h offset = 618h



MC_RGM_PRST2 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DSPI_12_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
3 LINFlexD_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
4 LINFlexD_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 LINFlexD_14_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

Table continues on the next page...

MC_RGM_PRST2 field descriptions (continued)

Field	Description
11 LINFlexD_16_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
12–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CAN_RAM_CTR_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 TTCAN_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 MCAN_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
26 MCAN_2_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
27 MCAN_3_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
28 MCAN_4_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

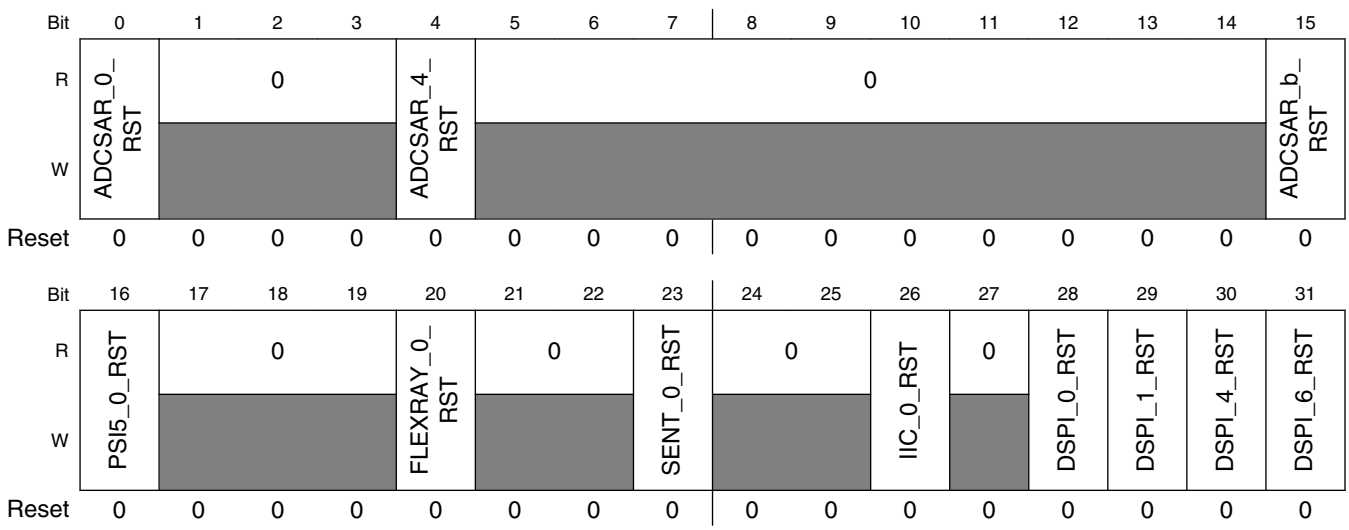
60.3.16 Peripheral Reset Register 3 (MC_RGM_PRST3)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 61Ch offset = 61Ch



MC_RGM_PRST3 field descriptions

Field	Description
0 ADCSAR_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ADCSAR_4_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_PRST3 field descriptions (continued)

Field	Description
15 ADCSAR_b_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
16 PSI5_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 FLEXRAY_0_ RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SENT_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 IIC_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 DSPI_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
29 DSPI_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

Table continues on the next page...

MC_RGM_PRST3 field descriptions (continued)

Field	Description
30 DSPI_4_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
31 DSPI_6_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

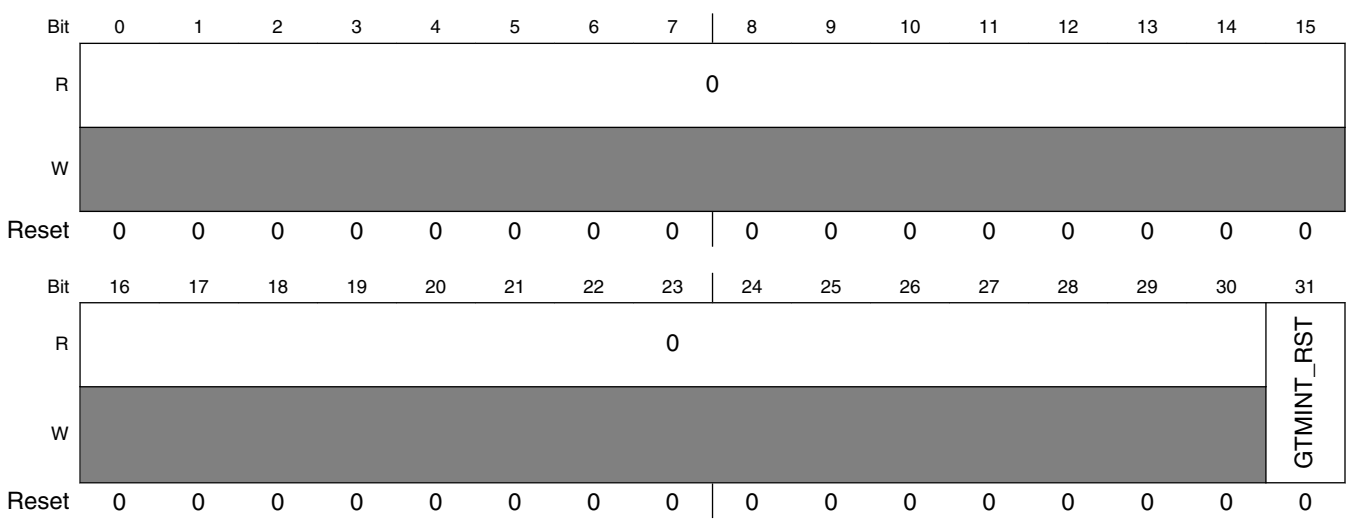
60.3.17 Peripheral Reset Register 4 (MC_RGM_PRST4)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 620h offset = 620h



MC_RGM_PRST4 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GTMINT_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

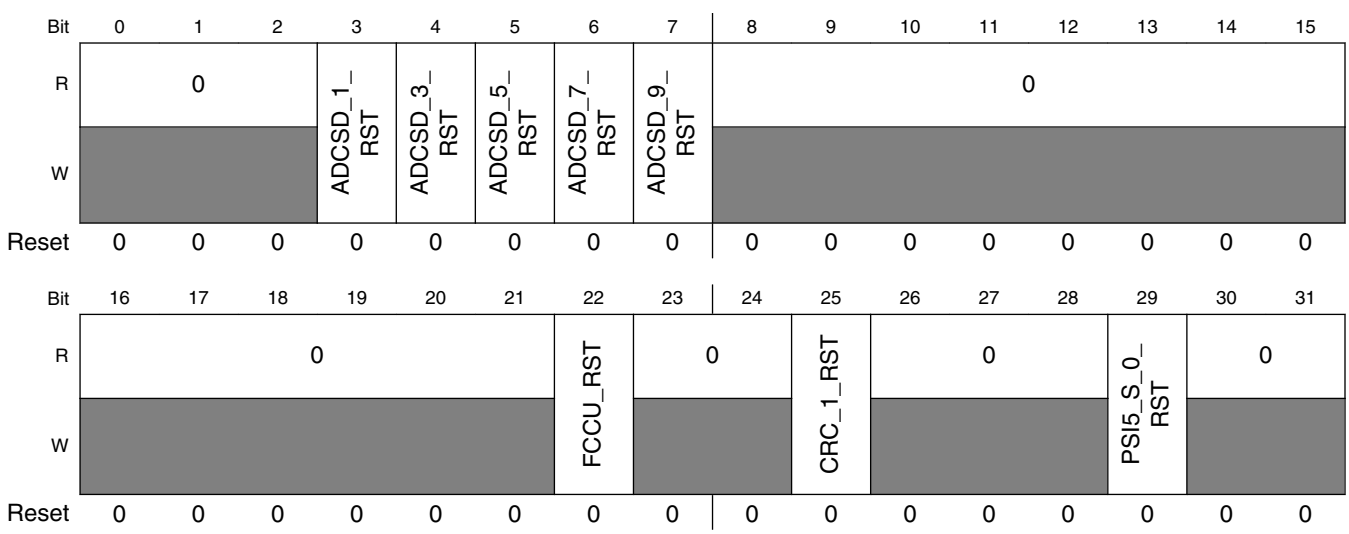
60.3.18 Peripheral Reset Register 5 (MC_RGM_PRST5)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 624h offset = 624h



MC_RGM_PRST5 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_PRST5 field descriptions (continued)

Field	Description
3 ADCSD_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
4 ADCSD_3_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5 ADCSD_5_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
6 ADCSD_7_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
7 ADCSD_9_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
8–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FCCU_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
23–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 CRC_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
26–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_PRST5 field descriptions (continued)

Field	Description
29 PSI5_S_0_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 0 forced reset of peripheral
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.3.19 Peripheral Reset Register 6 (MC_RGM_PRST6)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 628h offset = 628h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			LINFlexD_2_RST	0						LINFlexD_15_RST	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_PRST6 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_PRST6 field descriptions (continued)

Field	Description
3 LINFlexD_2_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
4–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 LINFlexD_15_ RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
11–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

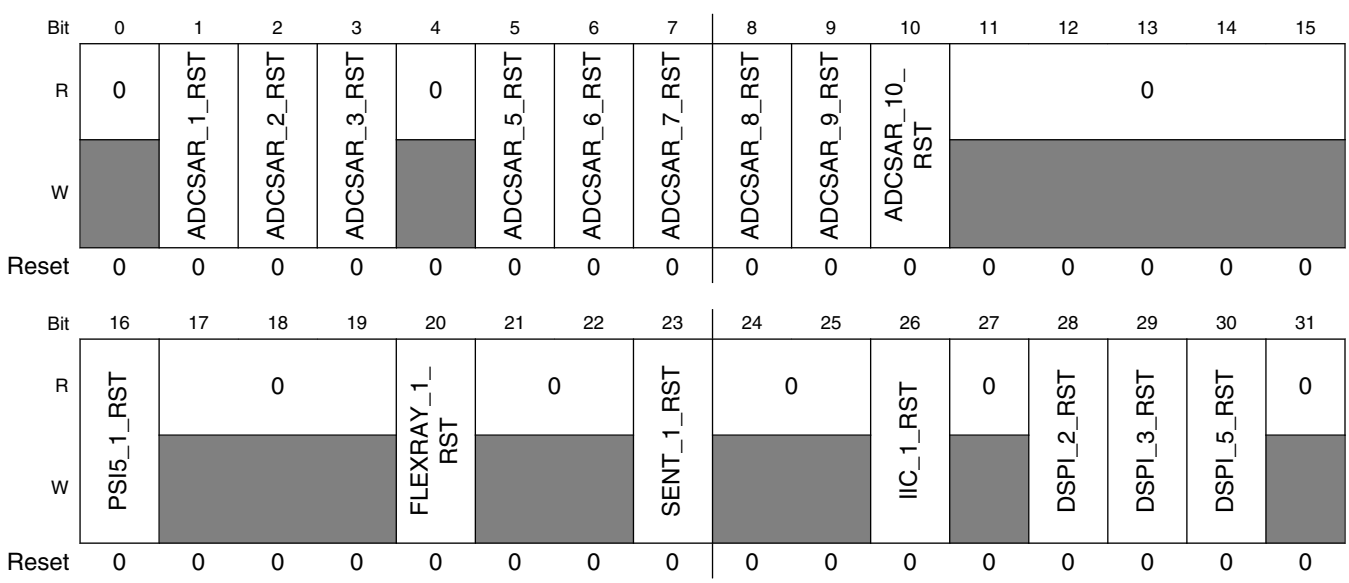
60.3.20 Peripheral Reset Register 7 (MC_RGM_PRST7)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Caution

Use the RGM_PRSTn registers with care. See 'Destructive' reset escalation for the proper sequence to prevent unexpected chip behavior.

Address: 0h base + 62Ch offset = 62Ch



MC_RGM_PRST7 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 ADCSAR_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
2 ADCSAR_2_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
3 ADCSAR_3_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 ADCSAR_5_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
6 ADCSAR_6_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
7 ADCSAR_7_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
8 ADCSAR_8_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
9 ADCSAR_9_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect.

Table continues on the next page...

MC_RGM_PRST7 field descriptions (continued)

Field	Description
	0 no forced reset of peripheral 1 forced reset of peripheral
10 ADCSAR_10_ RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
11–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 PSI5_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 FLEXRAY_1_ RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SENT_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 IIC_1_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 DSPI_2_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

Table continues on the next page...

MC_RGM_PRST7 field descriptions (continued)

Field	Description
29 DSPI_3_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
30 DSPI_5_RST	Peripheral reset - Writing a '1' to this bit will reset the corresponding peripheral. Writing a '0' to this bit will release the reset to the corresponding peripheral if the bit's current value is '1', otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.4 Functional description

60.4.1 Reset state machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in the following table.

Table 60-1. MC_RGM reset implications

Source	What Gets Reset	External Reset Assertion[1]	Boot Mode Capture
power-on reset	all	yes	yes
'destructive' resets	all except some clock/reset management	yes	yes
external reset	all except some clock/reset management and debug	programmable	yes
'functional' resets	all except some clock/reset management, STCU, FCCU, and debug	programmable[2]	programmable
shortened 'functional' resets	flip-flops except some clock/reset management, STCU, FCCU, and debug	programmable[2]	programmable[3]

1. 'external reset assertion' means that the ESR0 pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3
2. the assertion of the external reset is controlled via the RGM_FBRE register
3. the boot mode is captured if the external reset is asserted

Note

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in the following figure.

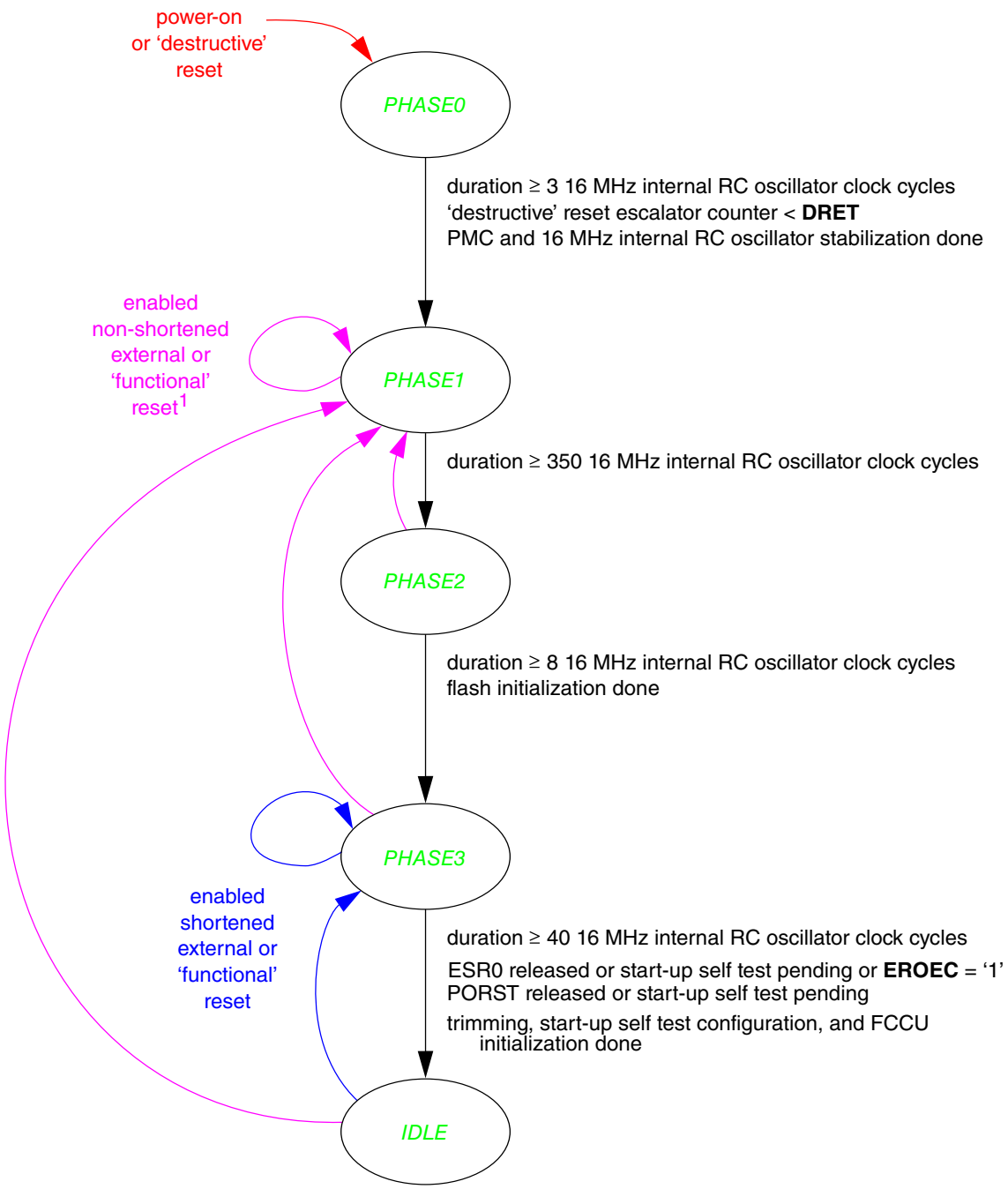


Figure 60-2. MC_RGM State Machine

60.4.1.1 PHASE0 phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- all enabled 'destructive' resets have been processed
- all processes that need to be done in PHASE0 are completed
    - PMC and 16 MHz internal RC oscillator stabilization
- a minimum of 3 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled 'destructive' reset event
- the 'destructive' reset escalator counter has not reached the value in the DRET field of the RGM_DRET resister

60.4.1.2 PHASE1 phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or 'functional' reset event if it has not been configured to trigger a 'short' sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- all enabled, non-shortened 'functional' resets have been processed
- a minimum of 350 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened 'functional' reset event

60.4.1.3 PHASE2 phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
    - flash initialization
- a minimum of 8 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

60.4.1.4 PHASE3 phase

This phase is a entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened 'functional' reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- all processes that need to be done in PHASE3 are completed
  - trimming, start-up self test configuration, and FCCU initialization
- a minimum of 40 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened 'functional' reset event
- ESR0 is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence or the EROEC bit of the RGM_EROEC register is '1'
- PORST is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

60.4.1.5 IDLE phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

60.4.2 'Destructive' resets

A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed. Exceptions to the latter are the software 'destructive' reset and the external power-on reset which do allow the memory content to be preserved if the chip is configured to not execute a self test on power-on, 'destructive', and external resets.

The status flag associated with a given 'destructive' reset event (RGM_DES.F_<destructive reset> bit) is set when the 'destructive' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'destructive' reset can be optionally disabled by writing bit RGM_DERD.D_<destructive reset>.

Note

The RGM_DERD register can be written only once between two 'destructive' reset events.

The chip's low-voltage detector threshold ensures that, when 'destructive' voltage out of range (LVD and/or HVD) is enabled, the supply is sufficient to have the 'destructive' event correctly propagated through the digital logic. Therefore, if a given 'destructive' reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given 'destructive' reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled 'destructive' reset will trigger a reset sequence starting from the beginning of PHASE0.

60.4.2.1 External power-on reset - PORST

The bidirectional external power-on reset pin PORST is a special case 'destructive' reset.

The detection of a falling edge on PORST will start the reset sequence.

The chip asserts PORST if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event

In this case, PORST is asserted until the end of PHASE0.

The PORST input is disabled immediately as of the PORST output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 µs after the PORST output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

In addition, the reset sequence is not exited into application until PORST is no longer driven low from off-chip. In the case the chip is configured to execute a self test on power-on, 'destructive', and external resets, this is the exit of PHASE3 after self test execution has completed.

60.4.3 External reset

The MC_RGM manages the external reset coming from ESR0. The detection of a falling edge on ESR0 will start the reset sequence.

The status flag associated with the external reset falling edge event (RGM_FES.F_ESR0 bit) is set when the external reset is asserted and the power-on reset is not asserted.

The external reset can optionally be disabled by writing bit RGM_FERD.D_ESR0.

Note

The RGM_FERD register can be written only once between two 'destructive' reset events.

An enabled external reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_ESR0 is set, the external reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an external reset event if configured via the RGM_FBRE register to assert the external reset
- a 'functional' reset event configured via the RGM_FBRE register to assert the external reset

In this case, ESR0 is asserted until all conditions for exiting PHASE3 have been met with the exception of the ESR0 assertion check.

If the OPT_RGM_ESR0_HW bit of the "UTEST Miscellaneous" DCF client is set to '0', the MC_RGM continues to assert ESR0 until the EROEC bit in the RGM_EROEC register has been cleared by software.

In addition, the MC_RGM asserts ESR0 while the start-up self test is being executed.

The ESR0 input is disabled immediately as of the ESR0 output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 µs after the ESR0 output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

If ESR0 is stopped being asserted by the MC_RGM and the 4 µs input mask period expires during the reset IDLE phase (e.g., due to the EROEC bit being cleared by software), and now a low value is detected on ESR0, an external reset falling edge event will not occur, and no new reset sequence will be triggered. For a new reset sequence to trigger, ESR0 must first be deasserted.

60.4.4 'Functional' resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (RGM_FES.F_<functional reset> bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'functional' reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note

The RGM_FERD register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a 'functional' reset. When RGM_FESS.SS_<functional reset> is set, the associated 'functional' reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a 'functional' reset should not reset the flash module.

60.4.5 Alternate event generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_D/FERD and RGM_D/FEAR registers as shown in the following table.

Table 60-2. MC_RGM Alternate event selection

RGM_D/FERD Bit Value	RGM_D/FEAR Bit Value	Generated Event
0	X	reset
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (i.e., at the reset source that caused the alternate request) and also clearing the appropriate RGM_D/FES status bit.

Note

SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request isn't asserted until the system clock is re-enabled.

Note

If a masked 'destructive' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored, and the MC_RGM will not send any SAFE mode/interrupt request to the MC_ME.

Note

If a masked 'functional' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any SAFE mode/interrupt request to the MC_ME.

60.4.6 'Functional' reset escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (i.e., entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a

write of any value to the RGM_FRET register and on any power-on or 'destructive' reset. If the counter reaches the value in the RGM_FRET register's FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

60.4.7 'Destructive' reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset which causes a reset sequence to be initiated (i.e., entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (i.e., entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

60.4.8 Individual Peripheral Resets

A peripheral may be reset individually without resetting the rest of the chip by setting the appropriate bit in the RGM_PRSTn registers. However, in order to prevent unexpected behavior by the chip, the following sequence must be observed by software.

1. Disable the peripheral that is to be reset via the MC_ME (see the MC_ME chapter for details).
2. Wait for the mode change to complete.
3. Set the bit corresponding to the peripheral in the RGM_PRSTn registers.
4. Wait at least two peripheral clock periods.
5. Clear the bit corresponding to the peripheral in the RGM_PRSTn registers.

The peripheral can be kept in reset indefinitely by not executing step 5. After the completion of step 5, the peripheral can be enabled again via the MC_ME.

Chapter 61
Boot Assist Flash (BAF)

61.1 Introduction

The MCU is booted through a collaboration of several blocks, hardware, and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, SSCM send reset vector to BAR, which sets PC to BAF start address.

The BAF code then checks the life cycle of the device. If the life cycle is FAIL_ANALYSIS, the BAF enters a loop in which it services the watchdog and CSE initialization is enabled. The BAF does not execute further since the read access to flash memory boot sectors for the core may be disabled if the device life cycle is FAIL_ANALYSIS. Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD or MCAN modules. The package pins/balls used by LINFlexD are the same used by the MCAN module. Hence, the external PHY can be either LIN or CAN, allowing the ASC@CAN operation (see Serial boot configuration for configuration details).

61.2 BAF image header

BAF image contains a 256-byte header starting from the BAF version number (see table "BAF memory map" table in the "BAF Configuration" section of this chapter). The header contains important information such as BAF version and production disable flag. The BAF header is explained in Table 61-1.

Table 61-1. BAF image header

Address offset	Size (bits)	Field description
0000h	32	BAF image version
0004h	32	Reserved
0008h	32	Reserved (part of Clock Jitter Activation Constant)
000Ch	32	Clock Jitter Activation Constant
0010h – 00FFh	Reserved	

61.2.1 BAF image version

The BAF version is a 32-bit field in the image header at the BAF starting address (see the "BAF memory map" table in the "BAF Configuration" section of this chapter). The 32-bit field is explained in Table 61-2.

Table 61-2. BAF image version

Field	Description
[0:7]	Major Number: This field contains the major version number for the BAF image
[8:15]	Minor Number: This field contains the minor version number for the BAF image
[16:23]	BAF Type: This field contains customer specific information.
[24:31]	Reserved

61.2.2 Clock Jitter Activation constant

The Clock Jitter Activation constant is a 32-bit flag, but the smallest element in flash that can be programmed is 64 bits, so the first 32 bits are reserved to allow programming of this constant. This 32-bit flag can be programmed by the customer and is used to activate the Clock Jitter feature that adds clock edge jitter to the baud rate clocks of several communications modules. As part of the BAF startup procedure, the BAF copies this constant from flash to the PASS Clock Jitter Enable register.

Table 61-3. BAF Clock Jitter Activation constant

Field	Description
[0:31]	Clock Jitter Activation Constant. The initial state of the production disable flag is erased (= FFFF_FFFFh). This equates to NO clock jitter.

61.3 Functional description

A functional description of the BAF modes is contained in the following sections.

61.3.1 Boot modes

Depending on the state and the content of its flash memory the device can enter one of three different boot modes:

- Boot from internal flash – If the internal flash contains application code with a valid boot header, the application is booted. The boot header contains flags that enable/ disable individual cores and their reset vectors.
- Serial boot – If no valid boot header is found and the current life cycle phase of the device is CUST_DEL (Customer Delivery) or JDP/FSL Production, an attempt is made to download the application by a serial protocol over LINFlexD or MCAN physical interface. The downloaded code is then executed by the initial boot core.
- Static mode – The device enters power safe mode and waits for an external reset.

61.3.2 Device initialization

The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact with the buddy device[1] and the Hardware Security Module (HSM) in the following way:

1. Buddy device: If the device is an emulation device (buddy device present) and the calibration RAM content is valid, then this RAM is automatically remapped over the calibration flash memory. This is intended to assist with cold-start calibration.
2. HSM: If the HSM is enabled and the "wait for HSM" flag is present, then the boot core waits for the HSM to become ready. It optionally executes a random instruction sequence to create random noise on the instruction bus while waiting.
3. BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an exception occurs during reading of the Boot header, the BAF ignores the ECC

1. "Buddy device" refers to a companion device that is used only for system development and debug activities. It is not used in production systems.

error and jumps to next boot header. If an ECC error occurs during reading of D-MEM, the BAF will still issue a destructive reset. BAF programs the Mode Entry module's MC_ME_MCTL[TARGET_MODE] to issue a destructive reset.

4. BAF uses memory of the Core_2 (IOP)Main core's D-MEM area as in the 'BAF memory map' table (at the beginning of this chapter) for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.

61.3.3 Flow of control

Figure 61-1 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

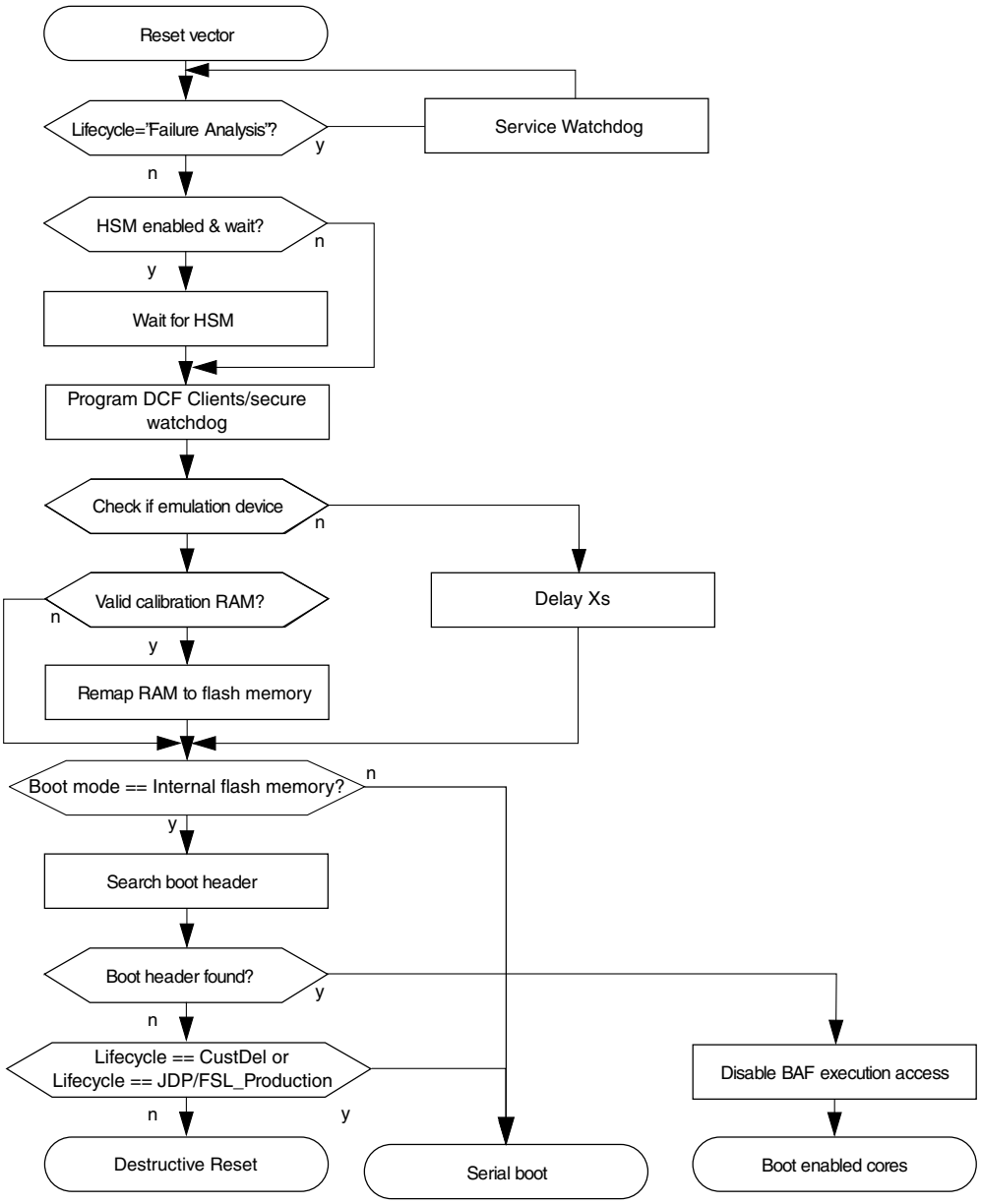


Figure 61-1. Flow of control

61.3.4 Optionally remap calibration RAM

If the device is an emulation device (buddy device present, SIUL2_MIDR1[ED] = 1), and the external overlay RAM has a valid content, then this RAM is automatically remapped over the selected flash memory regions. The details of the mapping are defined by a 400-byte configuration table at the beginning of the external overlay RAM. The layout of the table is depicted in Table 61-4.

The content of the overlay RAM is assumed valid if all of the following are true:

- The BD_INIT flag in the initialization status register of the buddy device (BD_SIUL2_ISR) is set.
- A CRC calculation over the configuration table yields the correct result.
- The version of the calibration data matches the version of the application software in flash memory (*((uint64_t*)ADDR_SWID) = DATID).

In this case, all 32 calibration remap descriptors (PFCRDs) are setup with the values contained in the configuration table. Finally, the PFCRDE field is copied to the enable register in the PFLASH controller.

Table 61-4. Remap configuration table

Address Offset	Name	Contents
00h	CRC_REF	CRC reference value
04h	ADDR_SWID	Address of 64 bit ID value in SW FLASH
08h	DATID	64 bit ID value of the calibration data
0Ch		
10h	PFCRDE	CRD enable flags
14h	LSTARTADDR0	PFCRD0 (logical start address)
18h	PSTARTADDR0	PFCRD0 (physical start address)
1Ch	CRDSize0	PFCRD0 (Master Mask, size)
20h	LSTARTADDR1	PFCRD1 (logical start address)
24h	PSTARTADDR1	PFCRD1 (physical start address)
28h	CRDSize1	PFCRD1 (Master Mask, size)
…	…	…
188h	LSTARTADDR31	PFCRD31 (logical start address)
18Ch	PSTARTADDR31	PFCRD31 (physical start address)
190h	CRDSize31	PFCRD31 (Master Mask, size)

The CRC calculation is performed using the CRC unit applying the CRC-32 (Ethernet protocol) polynomial. The calculation is initialized with a seed of 0000h. Then, the 100 32-bit words starting at table address offset 04h are moved to the CRC_INP register by the CPU. Finally, the content of the CRC_OUTP register is compared to the CRC_REF value.

61.3.5 Optionally wait for HSM

If the HSM is enabled (SSCM_UOPS[HSE] = 1), the boot core will need to wait for the HSM to become ready (if SSCM_UOPS[HSB = 1]). The core then polls the HSM until a ready flag is set. Both SSCM registers can be set by a DCF record. The boot core polls

the HSM to HOST FLAGS register (HSM_HSM2HTF) until its least significant bit is set. The boot core also services the watchdog while polling the HSM_HSM2HTF register. This is just in case the HSM is not ready in 16 ms, servicing the watchdog prevents the system going to reset.

61.3.6 Initialization of BAF DCF clients

61.3.6.1 Soft DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32-bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is FFFF_FFFFh). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be an intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF are listed in Table 61-5.

The DCF clients which are used for initialization of security watchdog are explained in Security watchdog configuration.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-bit, 16-bit and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

The serial boot DCF client provides a mechanism which can configure the BAF to perform serial boot or boot from internal flash. Details can be viewed in SER_BOOT_CBACK – BAF serial boot DCF client.

Table 61-5. BAF DCF client list

Address	Name	Label	Description
0000h	Security Watchdog Control register	SEC_SWT_CR	Data to write to SWT Control Register.
0001h	Security Watchdog Timeout register	SEC_SWT_TO	Security watchdog counter value.
0002h	Security Watchdog service address	SEC_IAC8	Instruction Address compare register value - Code present at this address services the watchdog.
0003h	Security watchdog CPU select	SEC_SWT_CPU	Determines which CPU is to use the security watchdog.
0004h – 01FFh	Reserved	—	Reserved for future use
0200h	Address	ADDR	The address to which DATA*n* is written.
0201h	32-bit data	DATA32	The 32-bit data that is written to ADDR.
0202h	16-bit data	DATA16	The 16-bit data that is written to ADDR.
0203h	8-bit data	DATA8	The 8-bit data that is written to ADDR.
0204h – 020Fh	Reserved	—	Reserved for future use.
0210h	Callback Address	CALLBACK	The address of a callback function.
0211h	Serial boot Callback address	SER_BOOT_CBACK	The address of serial boot callback function.
0212h – 0213h	Reserved	–	Reserved for future use.
0214h	Enable Calibration Delay Loop	CALIBRATION_PD_DELAY_ENABLE	Enables calibration remap equivalent delay loop on PD.
0215h	IVPR2 for CPU2	IVPR2_CPU2	Programs the IVPR2 for CPU2.
0216h – 7FFFh	Reserved	—	Reserved for future use.

NOTE

BAF boots with a watchdog timeout window set to 16ms.
Parsing more and more DCF Records may increase boot time.
But it should be taken care that number of DCF records should not be very high as it can cause watchdog window timeout.

61.3.6.2 Security watchdog configuration

BAF initializes the security watchdog before it jumps to the HOST CPU application. BAF implements four DCF clients for security watchdog initialization. The BAF parses the DCF records and looks for configuration data to be used to program the security watchdog registers. BAF first collects all information required to program the security watchdog from DCF records and then programs the security watchdog registers. BAF configures only SWT2 as security watchdog for the MainCPU2 (IOP) core. Configuration of the Security watchdog for other cores has to be performed by the application running on that core as BAF cannot program other core's IAC8 register. Security watchdog detailed configuration steps are explained in the section below.

61.3.6.2.1 Security Watchdog Control Register (SEC_SWT_CR)

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0000h while parsing, BAF programs the DCF client "Security watchdog control register" with the 32-bit value present in the DATA section of the DCF records. Once written, any subsequent writes to this DCF Client are ignored by BAF.

61.3.6.2.2 Security Watchdog Timeout (SEC_SWT_TO)

The BAF parses the DCF records. While parsing, if the BAF finds a DCF record with CS14 set and address field as 0001h, the BAF programs the security watchdog timeout DCF Clients. This is the security watchdog timeout register. The BAF checks the value obtained from DCF records. If the value yields a timeout greater than 30 seconds, then the BAF changes this value to 30 seconds. Once written, any subsequent writes to this DCF client are ignored.

61.3.6.2.3 Security Watchdog Service Address (SEC_IAC8)

BAF configures the security watchdog in 'fixed address execution service mode'. In this mode, the watchdog is serviced by executing code at the address loaded into the designated IAC8 (Instruction Address Compare) register. If BAF finds a DCF record

with CS14 set and address field as 0002h while parsing the DCF records, BAF programs the DCF clients for SEC_IAC8. This DCF client can only be written once. Subsequent writes are ignored.

NOTE

To program the core IAC8 special purpose register, the ownership of this register should be with software. If Hardware has its ownership software will not be able to write to it using mtspr instruction.

61.3.6.2.4 Security Watchdog CPU select (SEC_SWT_CPU)

This DCF client indicates to the BAF which Core will use the security watchdog. The BAF parses the DCF record and if it finds a record with CS14 set and an address field of 0003h, the BAF programs this DCF client. This means that the BAF uses the two least significant bits out of the 32-bit data in the DCF record to find the CPU number which will use the security watchdog. Table 61-7 and Table 61-6 shows the data layout for this client in the DCF record. This client may only be written once during reset. Subsequent writes are ignored.

NOTE

Due to the limitation that only the IAC8 on the CPU running the BAF code can be accessed, the BAF can only configure the Security Watchdog to operate on CPU2. If another CPU is selected, the Security watchdog is NOT configured.

Writing this client causes the BAF to configure the security watchdog based on data written to the three previous DCF clients. If any of the three previous DCF clients were not written, data is configured erroneously.

If security watchdog is assigned to CPU0 or CPU1, relevant CPU IAC8 register is programmed along with other debug configuration registers to enable IAC8 as watch point.

During normal BAF operation, SWT2 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). If the security watchdog is assigned to CPU2 (IOP), first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT2 and copied to SWT3. Then SWT2 is unlocked with current sequence and enabled as security watchdog.

During normal BAF operation, SWT2 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). To assign the security watchdog to the Main CPU, first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT2 and copied to SWT3. Then SWT2 is unlocked with current sequence and enabled as security watchdog.

Table 61-6. Security Watchdog Core select bit configuration

DCF Client Address 0003h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				0				0			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0	0	SC1	SC0

Table 61-7. Security Watchdog CPU select bit description

Field	Description
[31:2]	Reserved
SC[1:0]	These bits select the CPU which will use the security watchdog. 00 Reserved 01 Reserved 10 CPU2 (IOP Core) 11 Reserved

NOTE

BAF programs FCCU for enabling SWT2 and SWT3 as fault source and makes them hardware recoverable. This is done to ensure that fault latched in FCCU get cleared on reset and on next SWT timeout FCCU can again trigger reset. BAF does not clear the faults latched in FCCU as they can be of interest for application.

61.3.6.3 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0200h while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

Table 61-8. BAF DCF ADDR client configuration

DCF Client Address 0200h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDR[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADDR[16:31]															

Table 61-9. BAF DCF ADDR client description

Field	Description
ADDR[0:31]	32-Bit Address DCF client. This Address value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU

61.3.6.4 BAF DCF client - DATAX

CAUTION

While using DATAX DCF records, it should be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

61.3.6.4.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0201h while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2-bits to ‘0’ (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

Table 61-10. DATA32 DCF client configuration

DCF Client Address 0201h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table continues on the next page...

Table 61-10. DATA32 DCF client configuration (continued)

DATA[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA[16:31]															

Table 61-11. DATA32 DCF client description

Field	Description
DATA[0:31]	32-Bit DCF Data

61.3.6.4.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0202h while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to ‘0’ (Only 16-bit aligned writes are supported) and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

Table 61-12. DATA16 DCF client configuration

DCF Client Address 0202h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA[0:15]															

Table 61-13. DATA16 DCF client description

Field	Description
DATA[0:15]	32-Bit DCF Data

61.3.6.4.3 DATA8 – 8-bit data DCF client

The BAF parses the DCF records. During parsing, if the BAF finds a record with CS14 set and address field equal to 0203h, the BAF programs this 8-bit DCF client. The BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

Table 61-14. DATA8 DCF client configuration

DCF Client Address 0203h

Table continues on the next page...

Table 61-14. DATA8 DCF client configuration (continued)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved								DATA[0:7]							

Table 61-15. DATA8 DCF client description

Field	Description
DATA[0:7]	8-Bit DCF Data

61.3.6.5 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0210h while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Warning

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

61.3.6.6 SER_BOOT_CBACK – BAF serial boot DCF client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0211h during parsing, the BAF programs this serial boot DCF client. This DCF client is written with a 32-bit start address of a function which is invoked by the BAF. The BAF code checks to see if the start address is within the UTEST or BAF flash memory blocks,

then program flow is passed to this address if it is within either of these blocks. If the address is outside the the UTEST and BAF flash memory blocks, no function call is executed. The DCF client execution is aborted, and the BAF moves to the next DCF record. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in the BAF.

Warning

No parameters are passed to the function invoked by the BAF.

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a character value. The return value informs BAF if serial or normal internal flash boot should be executed (see Table 61-16 for details).

Table 61-16. Serial boot callback function return value definition

Return values	Description
00h	Boot from internal flash using standard search for boot record (see "Search boot header and boot options" at the beginning of this chapter).
01h	Serial boot using both MCAN and LINFlexD (in UART mode). See Optionally perform serial boot for details.
02h – FFh	Reserved

61.3.6.7 CALIBRATION_PD_DELAY_ENABLE – BAF calibration PD delay enable DCF client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0214h during parsing, the BAF programs this Calibration PD delay DCF client. This DCF client is written with a 32-bit key in DATA section of this DCF record. BAF supports Calibration Remap over external overlay ram on Emulation Device. On Production Device, this feature is not present but an equivalent delay loop can be activated. This is used if same BAF boot time is required on PD and ED. This delay can be activated using this DCF Client, if key present in DATA section of DCF record is 0x000055AA. For all other values, if DCF client itself is not present then delay is not activated. This DCF record can be programmed multiple times, but only the last instance of the DCF Record will be effective.

61.3.6.8 IVPR2_CPU2 - BAF set IVPR2 for CPU2 DCF client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0215h during parsing, the BAF programs the IVPR2 for CPU2 DCF client. This DCF client is written with a 32-bit value in DATA section of this DCF record. BAF parses this data value and programs it to IVPR2 for CPU2 just before jumping to application. If this DCF client is not present, then IVPR2 will contain address of BAF vector table. This DCF record can be programmed multiple times, but only the last instance of the DCF Record will be effective.

61.3.7 Production disable activation protocol and implementation

Module failure analysis is a standard requirement, but some OEMs have an additional requirement that upon entering failure analysis mode the MCU operation is modified in such a way that the MCU can never be used in a production ECU. The chosen operational modification for production disable is to apply clock jitters to can modules so that it prevents active communications with another can device on a can network (see the PASS chapter for details).

The BAF maintains a 64 bit flag in BAF code flash region as explained in Clock Jitter Activation constant. On every POR the BAF copies this clock jitter activation constant to the Clock Jitter Enable register in the PASS module to enable jitter on the MCAN baud clock. Default value of this flag is shown in Table 61-3. Hence, jitter on the MCAN baud clock is disabled if the default value is used. To enable jitter on CAN clock the value of this flag should be modified to FFFF_FFFEh.

The BAF flash memory is write protected and locked. To allow modification of the Clock Jitter flag and activate the production disable feature, the BAF implements production disable activation protocol as explained in Table 61-18. The BAF implements this protocol in a Power.org EABI compliant callback routine. This callback routine can be invoked by programming a callback based soft DCF client as explained in CALLBACK – BAF callback DCF client. The callback function address is show in Table 61-17.

Table 61-17. Production Disable Callback routine entry point

Description	Address
Production disable protocol callback function address	0040_40E0h
Production Disable falsh Driver SRAM area	5280_0408h - 5280_0808h

Table 61-18. Production Disable protocol description

Step	Description	Tool action or response	MCU action or response
1	After power-on tool holds the MCU in reset and writes the activation code to JDC via JTAG interface. The tool pulls hardware interlock pin high, then releases reset.	Assert hardware interlock pin Write 0xF5A0_AA55 to JDC	Starts production disable activation
2	The MCU responds by writing to the JDC an acknowledgment code and which of the 4 PASS passwords are required to unlock the BAF flash block	—	Write FA50_000*n*h to JDC if password for pass group n is required to unlock the BAF flash block. (n is either 0,1,2 or 3)
3	The tool reads the acknowledge from JDC module and writes the relevant password to the JDC module. MCU performs password comparison and writes error code to JDC module password provided by tool was incorrect.	Read JDC – Write 256 bit password for pass group n to JDC in eight 32 bit words. (n is 0,1,2 or 3)	Write FA50_F00*n*h if password does NOT match. Once the status has been read by the tool then cause reset. (n is either 0,1,2 or 3)
4	If more than one password is required to unlock the BAF flash block step 2 and step 3 are repeated until all relevant password has been supplied.	—	If password matches step 2 and step 3 are repeated for the next pass group until all pass group that lock BAF flash has been processed.
5	The MCU then attempts to program the can jitter activation constant within the BAF memory block. The can jitter activation constant flag is a write once flag. Once written it cannot be re-programed as BAF memory block is OTP. MCU write a code to JDC to signal success or failure of flash programming operation and issue reset.	—	Program can jitter activation constant in BAF flash block. If successful write FA51_0000h to JDC. If unsuccessful write FA51_F000h to JDC Issue reset.

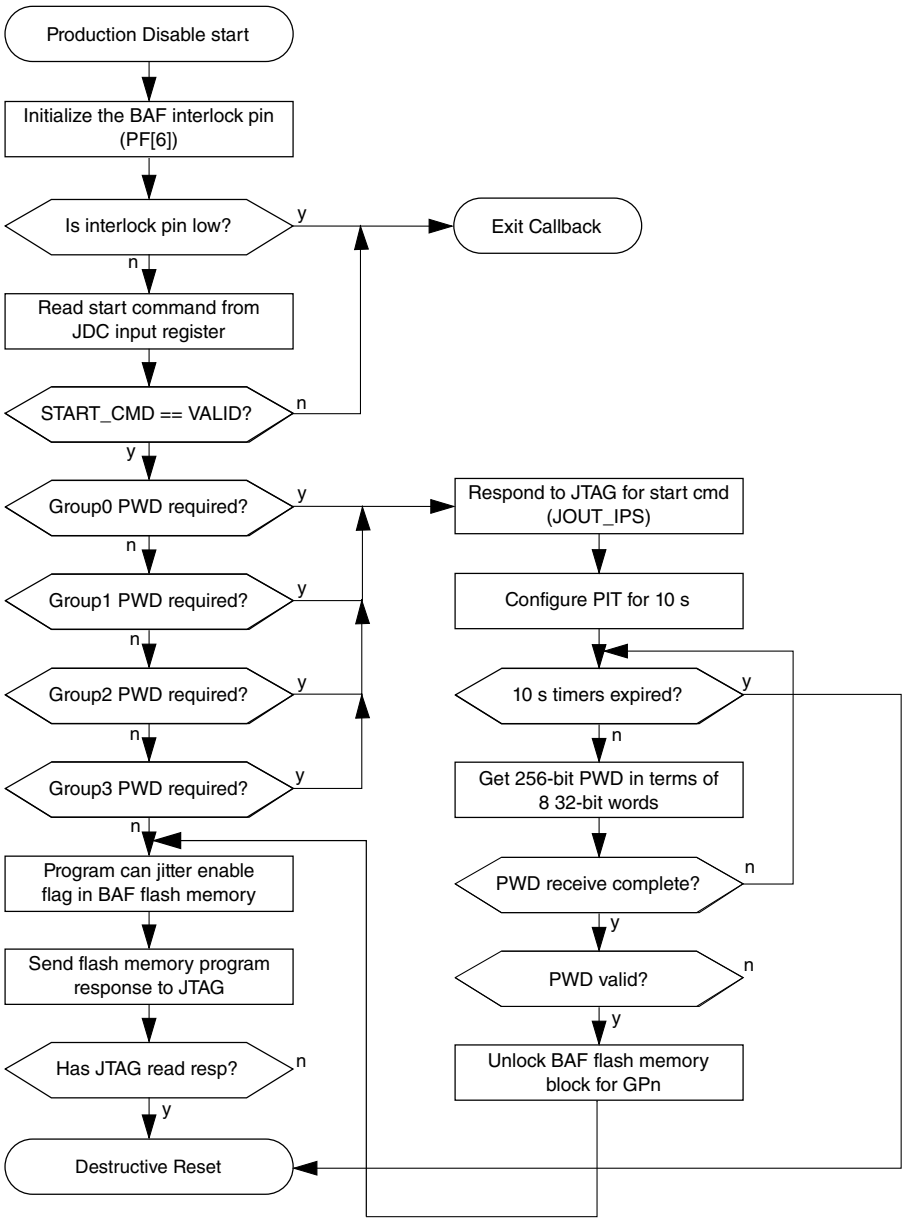


Figure 61-2. Production Disable callback function flow chart

61.3.8 TDM Diary Operation

On every reset, BAF copies the first four bytes of last record (8-byte) of every Tamper Detect Region to corresponding Software Tamper Override Key Region Registers in TDM. BAF reads the base address of TDM Diary (group of 6 continuous Tamper Detect Regions, 2k each) from DBA register in TDM. For more details on TDM please refer to corresponding section. This is done as a part of TDM Diary Override enhancement.

61.3.9 Optionally perform serial boot

If no boot header is found in any of the locations mentioned above, the BAF determines the Life Cycle status of the device. If the Life Cycle is in CUST_DELIV (Customer Delivery) or JDP/FSL Production, it attempts a serial boot. Otherwise, the boot has failed and BAF issues a destructive reset.

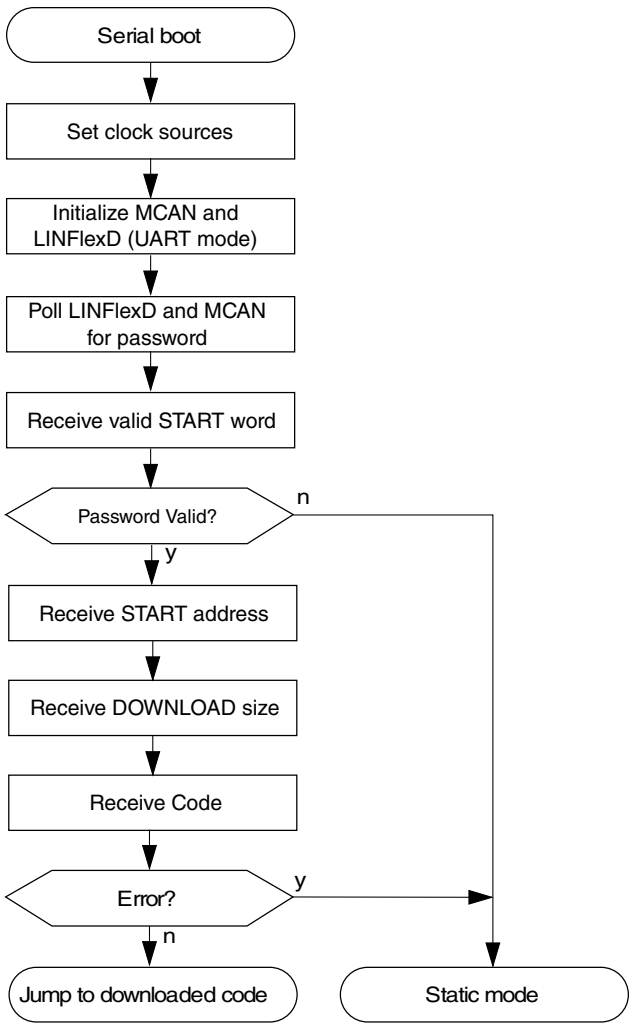


Figure 61-3. Flow of control during serial boot

The clock source in the MC_CGM module is configured and the PADs are set up for LINFlexD and MCAN signals (see Serial boot configuration for details), then the code is downloaded using LINFlexD or MCAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via the LINFlexD or MCAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process, both LINFlexD and MCAN are configured in passive listening mode, connected to the same pair of pads. As soon as the interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The interface that is not receiving is switched back to its reset state and does not participate further in the serial boot process.

If a CAN message of 8 bytes with ID 11h is received on the or MCAN controller first, the BAF program:

- Disables LINFlexD.
- Configures LINFlexD RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message is received that does not contain 8 bytes or does not have an ID 11h by the MCAN controller, or the MCAN controller generates a CAN error, it is probable that the tool is sending a LIN frame. Under such circumstances the BAF program:

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error.
- Continue monitoring for valid CAN messages.

If any byte from LINFlexD is received first, the BAF program:

- Checks if the first byte is FEh of the password. If not, continue polling. The data may be a valid CAN message.
- If it is first byte of the password (FEh) then disable MCAN controller and restore to its reset state.
- Configures MCAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

61.3.9.1 Serial boot mode protocol

From a high-level perspective, the download protocol over CAN and LIN interfaces follows steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes and VLE bit to MCU.
3. Host transmits a sequence of data bytes that are to be executed.
4. CPU2 executes code from start address provided in 2

Each step must be complete before the next step starts.

The exact protocol between LinFlex and MCAN differs slightly.

61.3.9.2 LINFlexD download protocol

The communication is performed in half-duplex mode, Any transmission from the host is followed by the MCU transmission:

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received data to the host.
3. The host verifies whether the echo is correct.
   - If the data is correct, the host can send the next byte of data.
   - If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded and the VLE bit.
4. x bytes of code that is downloaded to SRAM and executed, (x must match the value specified in 3).

61.3.9.3 MCAN download protocol

The MCAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 11h as the ID and 8-byte length to send the password. The MCU transmit the same message but with and ID of 1h.
2. A message with 12h as the ID and 8-byte length to send the start address, size of the code and VLE mode bit. The MCU transmit back the same data but with an ID of 2h.
3. A message with 13h as the ID is used to send data that is to be downloaded to SRAM. The MCU transmit the same data with an ID of 3h.

61.3.9.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password FEED_FACE_CAFE_BEEFh.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode(low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

61.3.9.5 Download start address, VLE bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bit and 31-bit code length as shown in Table 61-19.

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and BookE Code (VLE = 0).

The MCU only supports VLE code. If the BAF detects that the host is sending Book-E code, the BAF puts the device into static mode.

The start address defines where the received data is stored and where the MCU branches after the download is finished. BAF ignores the three LSB bits of download address to avoid ECC on download address. But BAF does not align the address while jumping to application for executing. So, download address should be passed as 64-bit aligned for proper execution.

The Length defines how many data bytes have to be loaded.

Table 61-19. Start address, VLE bit and download size in bytes

Bytes 0-3															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
START_ADDRESS[31:16]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
START_ADDRESS[15:0]															
Bytes 4-7															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VLE	CODE_LENGTH[30:16]														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Table 61-19. Start address, VLE bit and download size in bytes (continued)

CODE_LENGTH[15:0]

61.3.9.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until the number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-bit boundary, the BAF fills it with zero's and renders the data aligned on a 64-bit boundary. After the last data is written to SRAM, the BAF writes a dummy double-word (0000_0000_0000_0000h) to the next address location. This is written to avoid possible ECC error during core prefetch.

61.3.9.7 Execute code

The BAF program waits for the last echo message transmission being completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and MCU is controlled by new code executing from SRAM.

61.3.10 Serial boot configuration

61.3.10.1 Clock Settings

NOTE

XOSC load capacitance can be selected as internal via UTEST_MISCELLANEOUS dcf record. On such virgin device where no dcf record is programmed, XOSC may be unstable because of resulted wrong load capacitance. As CAN_CLK

should be derived from XOSC, CAN boot can not be performed on such designs. LinFlex boot can be performed on such cases because LIN_CLK can be derived from IRC.

Serial boot interfaces(LIN and MCAN) derives its clock requirements from PBRIDGE_CLK, CAN_CLK and LIN_CLK. LIN_CLK can be derived from both IRC or XOSC for successful operation. But in case of MCAN, only XOSC can be selected as CAN_CLK because of high precision requirements. Also in case of MCAN boot, module clock (PBRIDGE_CLK) should be more than Protocol Clock (CAN_CLK). For LINFlex, LIN_CLK and PBRIDGE_CLK should satisfy a relation of "2/3 × LIN_CLK ≥ PBRIDGEx_CLK > 1/3 × LINCLK". As XOSC can vary from 8 MHhz to 44 MHz, PBRIDGE also should be more than or equal to 44 MHz. To achieve the clock requirements, PLL0 is locked over IRC with a multiplying factor of 5.5 and SC_DC2 divider set to 2. PLL0_PHI is selected as system clock source via MC_ME. This makes system and PBRIDGE clock at 44 MHz. LIN_Clk is also selected as PLL0_PHI (88MHz). Also, during serial boot all peripherals are enabled. After downloading application code, clock settings are not restored and no peripheral is disabled. To change the clock source, first all peripherals must be disabled and selectors in CGM must be restored.

BAF enables XOSC via MC_ME but does not wait for M_TRANS to go low because in case of faulty XOSC, mode transition may not complete. BAF waits for 3 ms to provide enough time for XOSC to becomes stable. If XOSC is still unstable then CAN boot is not performed. Some reasons of XOSC being unstable are XOSC gets disconnected or wrong value of internal load capacitance in case of internal load capacitance option is selected in UTEST_MISCELLANEOUS dcf record.

61.3.10.2 LINFlexD configuration

Boot using UART protocol is implemented by the LINFlexD module.

NOTE

See the pin configuration table at the beginning of this chapter for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx pin is configured as an input and muxed to the LINFlexD module. The LINFlexD_Tx pin remains in a passive input state until the first byte of protocol is received. It is then configured as an output and muxed to LINFlexD module. The application may not be using the Serial Boot Mode function, and may be using the pin used by LINFlexD_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data (such as LIN or CAN). The side effect of this is that all data transmitted on the Tx pin is also received on the Rx pin. To prevent the LINFlexD module from receiving its own transmission, the Rx pin is blocked during transmission.

The LINFlexD controller is configured to operate at a baud rate of 250 Kbps, using an 8-bit data frame with no parity bit and 1 stop bit.

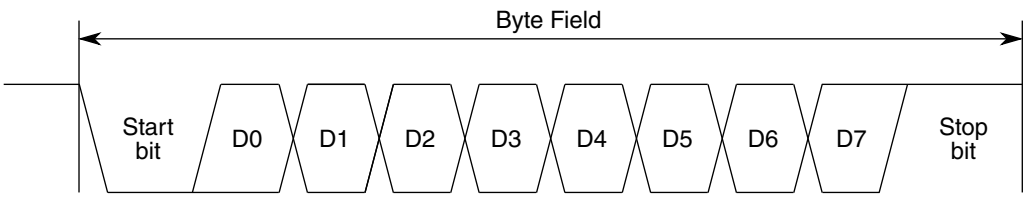


Figure 61-4. LINFlexD bit timing in UART mode

61.3.10.3 MCAN configuration

The MCAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0B specification (see Table 61-20 for baud rates).

NOTE

See the pin configuration table at the beginning of this chapter for pins used by the MCAN module.

The Bit timing is configured as shown in Figure 61-5.

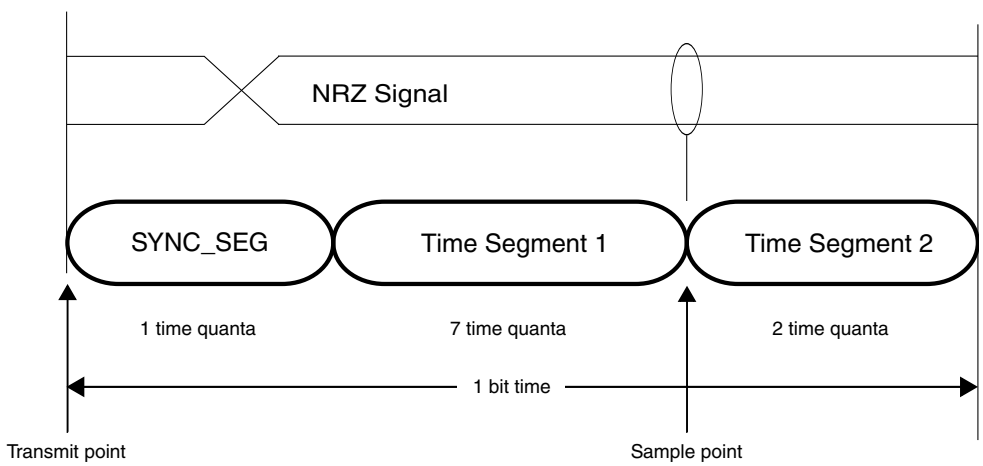


Figure 61-5. MCAN bit timing

61.3.10.4 LINFlexD and MCAN pin configuration

Please see the BAF configuration at the beginning of this chapter for pin details.

61.3.10.5 Baud rates

The LINFlexD baud rate is set to 250 Kbps. Table 61-20 shows some possible baud rates.

The MCAN baud rate is set by the frequency of the external crystal / 40.

Table 61-20. LINFLexD and MCAN baud rates

Crystal frequency	LINFlexD baud rate	MCAN baud rate
8 MHz	250 Kbps	200 Kbps
12 MHz	250 Kbps	300 Kbps
16 MHz	250 Kbps	400 Kbps
20 MHz	250 Kbps	500 Kbps
40 MHz	250 Kbps	1000 Kbps

61.3.10.6 Protocol summary

Table 61-21 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 61-21. LINFlexD serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + 31 bits of download data size in bytes (MSB first)	VLE bit + 31 bits of download data size in bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	—	8 × 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	none	none	Branch to downloaded code

Table 61-22 shows the MCAN serial boot protocol.

Table 61-22. MCAN serial boot mode download protocol

Protocol step	Host message sent	BAF response message	Action
1	CAN ID 011h + 64-bit password	CAN ID 001h + 64-bit password	Password checked against fixed password (FEED_FACE_CAFE_BEEFh). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 012h + 32-bit store address + 32-bit (1 VLE bit, and 31 bits of download data size in bytes)	CAN ID 002h + 32-bit store address + 32-bit (1 VLE bit, and 31 bits of download data size in bytes)	Load address and size of download are stored for future use.
3	CAN ID 013h + 8 to 64 bits of raw binary data	CAN ID 003h + 8 to 64 bits of raw binary data	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAF returns IO pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2)
5	none	none	Branch to downloaded code

61.4 Resources

- Clocks
  - Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).
- LINFlexD and MCAN
  - The LINFlexD and MCAN modules are used if serial boot mode is entered. This can be through failing to find a valid boot header, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.
- CRC
  - The CRC module is used to validate external overlay contents, if and only if the device is an emulation device and the buddy device has been initialized.
- PFLASH
  - PFLash module is used to disable BAF execution.
- Interrupts

    - No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.
- Security watchdog
    - When SWT2 is chosen as security watchdog, SWT3 is used for task monitoring purpose.
- FCCU
    - FCCU is used to configure Fault Reaction of SWT2 and SWT3.
- TDM
    - TDM for implementing TDM Diary enhancements.

Chapter 62
System Status and Configuration Module (SSCM)

62.1 Introduction

62.1.1 Overview

The System Status and Configuration Module (SSCM) is pictured in the figure below.

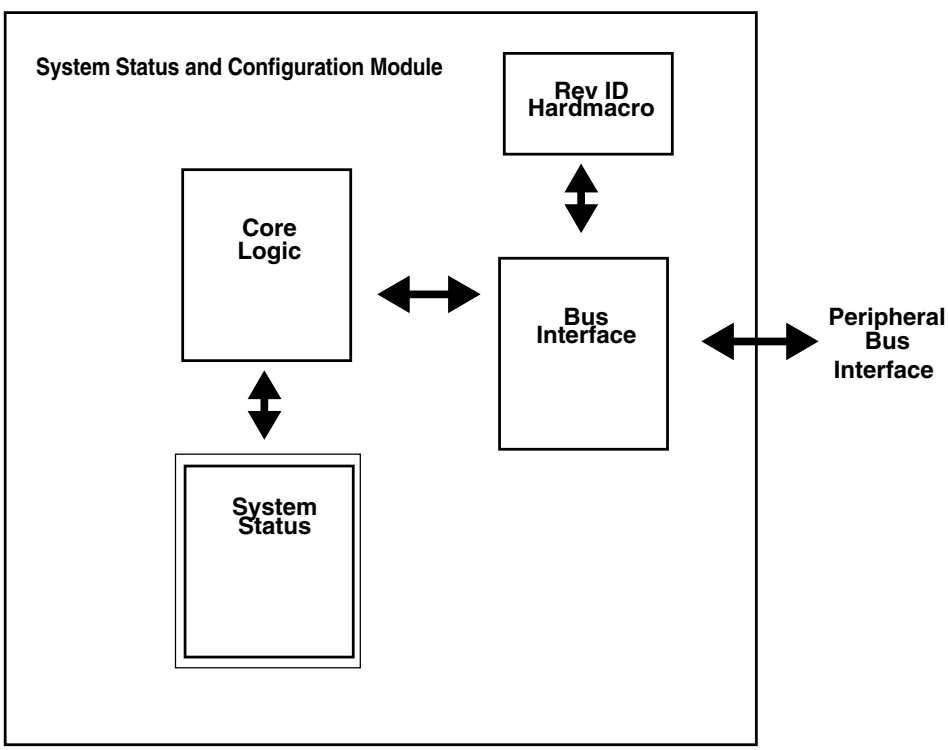


Figure 62-1. SSCM block diagram

62.1.2 Glossary

The table below shows a glossary of terms used through out the text.

Table 62-1. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

62.1.3 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine primary boot vector
  - Determine HSM boot vector
  - Decodes the Device Life Cycle
- Bus abort enable/disable
- Peripheral abort enable/disable

62.1.4 Modes of operation

The SSCM operates identically in all system modes.

62.2 External signal description

The SSCM has no external pins.

62.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

The table below shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

NOTE

All registers are accessible via 8-bit, 16-bit or 32-bit reads and/or writes. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit read or write, SSCM Base + 02h, but performing a 16-bit access to SSCM Base + 0003h is illegal.

NOTE

SSCM does not support transfer errors in memory holes.

SSCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SSCM System Status (SSCM_STATUS)	16	R	See section	62.3.1/3324
2	SSCM System Memory and ID Register (SSCM_MEMCONFIG)	16	R	See section	62.3.2/3325
6	SSCM Error Configuration Register (SSCM_ERROR)	16	R/W	0000h	62.3.3/3326
C	Password comparison register low word (SSCM_PWCMPH)	32	W (always reads 0)	0000_0000h	62.3.4/3327
10	Password comparison register low word (SSCM_PWCMPL)	32	W (always reads 0)	0000_0000h	62.3.5/3327
20	SSCM HSM and User Option Status Register (SSCM_UOPS)	32	R	See section	62.3.6/3328
28	Processor Start Address Register (SSCM_PSA)	32	R	See section	62.3.7/3329
30	SSCM HSM Start Address Register (SSCM_HSA)	32	R	See section	62.3.8/3329
34	Life Cycle Status Register (SSCM_LCSTAT)	32	R	See section	62.3.9/3330

62.3.1 SSCM System Status (SSCM_STATUS)

The System Status register reflects the current state of the system.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0	CER	0	NXEN1	NXEN	1	Reserved	0
Write		w1c						
Reset	0	0	0	*	*	1	*	0

Bit	8	9	10	11	12	13	14	15
Read	BMODE			VLE	0	0	0	
Write								
Reset	*	*	*	*	0	0	0	0

* Notes:
- VLE field: Reset value depends on the associated option bits in flash memory.
- BMODE field: Reset value depends on the device status after leaving reset.
- Reserved field: This field can reset to 0 or 1.
- NXEN field: Reset value depends on the device status after leaving reset.
- NXEN1 field: Reset value depends on the device status after leaving reset..

SSCM_STATUS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error. If a non-permanent error is detected, this bit can be cleared by writing a 1. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 NXEN1	Processor 1 Nexus enable status 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
4 NXEN	Processor 0 Nexus enable status 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled

Table continues on the next page...

SSCM_STATUS field descriptions (continued)

Field	Description
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This field is for internal use. Users should ignore read values and should not attempt to write the field.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 BMODE	Device Boot Mode This field is only updated during reset. 101 Boot from BootFlash
11 VLE	Variable Length Instruction Mode When booting in SC mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. **NOTE:** Reset value depends on the associated option bits in flash memory. 0 SC Boot Location contains standard PPC code 1 SC Boot Location contains VLE code
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	JPIN										1	MREV				0
Write																
Reset	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	0

* Notes:
- MREV field: Reset value can be found in the SSCM Configuration section.
- JPIN field: Reset value reflects the JTAG ID of the device.

SSCM_MEMCONFIG field descriptions

Field	Description
0–9 JPIN	JTAG Part ID Number **NOTE:** Reset value reflects the JTAG ID of the device.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11–14 MREV	Minor Mask Revision
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.3 SSCM Error Configuration Register (SSCM_ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0														PAE	RAE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSCM_ERROR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

14
PAE
	Peripheral Bus Abort Enable.
	
	This bit enables bus aborts on:
	• any access to a peripheral slot that is not used on the device.
	  • This feature is intended to aid in debugging when developing application code.
	• illegal accesses to off-platform peripherals.
	  • On platform peripherals cannot be controlled by SSCM peripheral abort.
	
	0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception
	1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception

15 RAE	Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. **NOTE:** Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the AIPS level). In this case, the PAE and RAE register bits have no effect on the abort.

Table continues on the next page...

SSCM_ERROR field descriptions (continued)

Field	Description
	0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception

62.3.4 Password comparison register low word (SSCM_PWCMPH)

These registers allow to unsecure the device, if the correct password is known. They are intended for device-internal operation only.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PWD_HI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSCM_PWCMPH field descriptions

Field	Description
0–31 PWD_HI	Upper 32 bits of the password.

62.3.5 Password comparison register low word (SSCM_PWCMPL)

These registers allow to unsecure the device, if the correct password is known. They are intended for device-internal operation only.

Address: 0h base + 10h offset = 10h

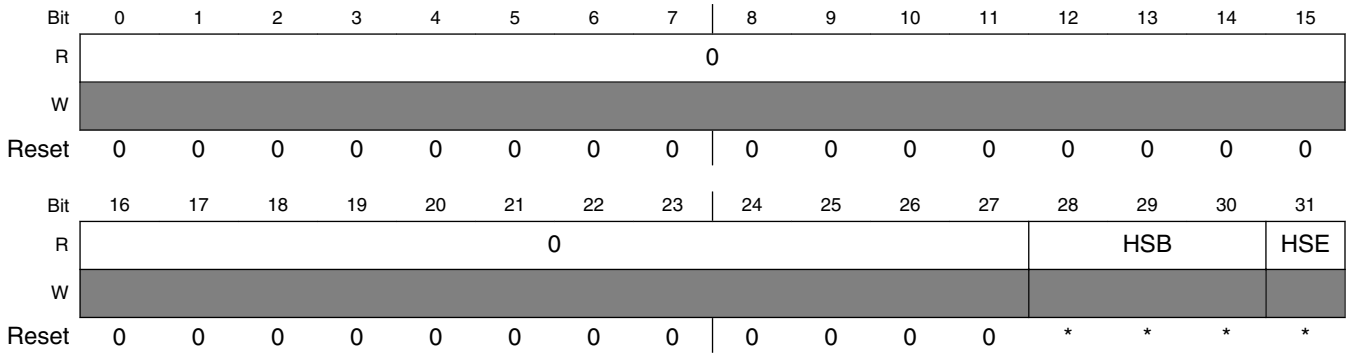
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PWD_LO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSCM_PWCMPL field descriptions

Field	Description
0–31 PWD_LO	Lower 32 bits of the pasword

62.3.6 SSCM HSM and User Option Status Register (SSCM_UOPS)

Address: 0h base + 20h offset = 20h



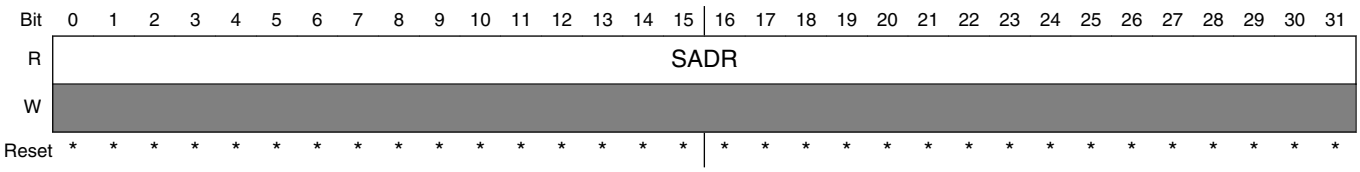
* Notes:
- HSE field: Values are determined by DCF record and life cycle - default is 0.
- HSB field: Values are determined by DCF record and life cycle - default is 0.

SSCM_UOPS field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HSB	HSM Boot Configuration These bits can be set via DCF record (see HSM documentation). All combinations not listed below are reserved. 000 When the device is in BAF mode, BAF will not wait for HSM ready flag before allowing application code to start. In SC mode the SSCM will not wait for HSM ready flag before allowing application code to start. 001 When the device is in BAF mode, BAF waits for HSM ready flag before allowing application code to start. In SC mode the SSCM waits for HSM ready flag before allowing application code to start.
31 HSE	HSM Enabled HSE is set to 1 if HSM is enabled via DCF record

62.3.7 Processor Start Address Register (SSCM_PSA)

Address: 0h base + 28h offset = 28h



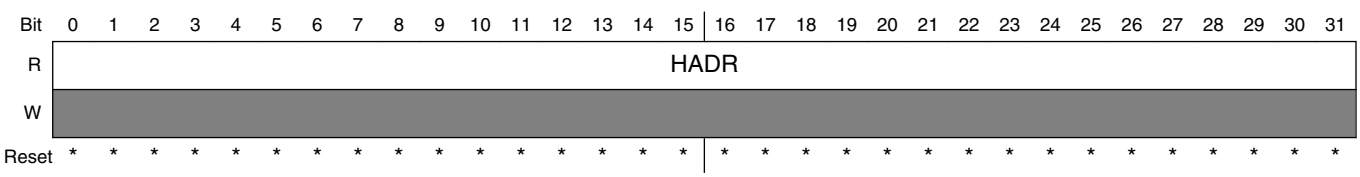
* Notes:
- SADR field: Reset value is the start address

SSCM_PSA field descriptions

Field	Description
0–31 SADR	Processor Start Address The boot processor will start executing application code from this address.In SC mode this indicates the location determined by the RCHW search. In BAF mode this will show the start address of the BAF code.

62.3.8 SSCM HSM Start Address Register (SSCM_HSA)

Address: 0h base + 30h offset = 30h



* Notes:
- HADR field: HSM Start Address

SSCM_HSA field descriptions

Field	Description
0–31 HADR	HSM Start Address HSM will start executing application code from this address. The value of this register is determined by the HSM boot header search.

62.3.9 Life Cycle Status Register (SSCM_LCSTAT)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						Reserved		0					LC		
W																
Reset	0	0	0	0	0	0	*	*	0	0	0	0	0	*	*	*

* Notes:
- LC field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- Reserved field: Reset Value is '11' for a valid secured part.

SSCM_LCSTAT field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LC	Life Cycle Determined by the SSCM by reading the LC slots. The encoding is: 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 JDP Production 111 In Field

62.4 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

62.4.1 ECC error monitoring

During flash memory scanning (such as Life Cycle or DCF records), the SSCM will monitor the flash memory status signals for ECC and other unexpected access errors. If such an error occurs, the SSCM will:

- Request a destructive reset

NOTE

1. Signal the FCCU or MEMU.
2. For differential region, if any valid DCF records are programmed then there will be checked for ECC and stop record for DCF would not be checked for ECC.

62.4.2 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Boot Assisted by Flash (BAF) - The device will boot from UTest NVM Flash Block, (interface with the HSM if available) and perform device setup.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location) then the device will enter static mode.

If the HSM is enabled via DCF record, the HSM will boot from the location found via the HSM boot header search. HSM boot header search.

62.4.3 BAF configuration

By default the device will boot from BAF.

In BAF mode (SSCM_STATUS[BMODE]=101b), the boot core will start executing BAF code at the location shown in the configuration section of the “Boot Assist Flash (BAF)” chapter.

62.4.4 HSM boot header search

The SSCM searches the code flash memory of the HSM module for a valid boot header (see the SSCM section of the “Device Configuration” chapter for address(es)) and provides the boot vector to the CPU of the HSM module.

The HSM Boot Header has the format as described in the table below.

Table 62-2. Boot header structure

00h	Boot Header ID	
08h	Reserved	Start Address
10h	Reserved for Configuration Bits	

A valid Boot Header ID has the value FFFF_0000_FFFF_0000h.

The first valid Boot Header found is the one which is passed to the HSM CPU. In order to switch from one Boot Header to a Boot Header at a higher address, the Boot Header at the lower address must be invalidated. This can be achieved by over-programming the Boot Header ID location with 0000_0000_0000_0000h. Since this value results in the same ECC encoding it will still be possible to read the location without causing an ECC error.

62.4.5 Life Cycle

The SSCM will determine the Life Cycle (LC) of the device by reading the LC slots from the UTEST flash memory. The read operation is done during the reset phase with normal timings and it is protected by operating monitors and ECC check. In addition, a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- JDP Production
- Customer Delivery
- OEM Production
- In Field
- Fail Analysis

The LC is written into 5 slots, 128 bits each, at fixed locations in the UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- the valid field
- the invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 statuses as seen in Table 62-3.

Table 62-3. LC slot status

LC slots		LC slot value
Valid field (64 bits)	Invalid field (64 bits)	
Erased	Erased	Erased
Marked	Erased	Active
Marked	Marked	Inactive
Any other values		Illegal

"Marked" in this case means that the value has been programmed with the bit pattern 55AA_50AF_55AA_50AFh. "Erased" is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. Table 62-4 shows how the slots are arranged in memory.

Table 62-4. LC slots in memory

Offset	LC slot word
00h	Valid[31:0]
04h	Valid[63:32]
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in Table 62-5. The priority of the entries is from top to bottom, so the first row that applies determines the resulting Life Cycle.

NOTE

"Erased" means the slot returns all 1s.

Table 62-5. LC slots

LC slot 0 JDP Production	LC slot 1 Customer Delivery	LC slot 2 OEM Production	LC slot 3 In Field	LC slot 4 Failure Analysis	Resulting life cycle
Active	Erased	Erased	Erased	Erased	JDP Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	Inactive	**Active**	Erased	Erased	OEM Production
Inactive	Inactive	Inactive	**Active**	Erased	In Field
Inactive	Inactive	Inactive	Inactive	**Active**	Failure Analysis
Erased	Erased	Erased	Erased	Erased	System Reset
Invalid[1]	Invalid	Invalid	Invalid	Erased	In Field
Invalid	Invalid	Invalid	Invalid	Not Erased	System Reset

1. Any value programmed other then shown in table above for Erased, Active and Inactive would be treated as illegal.

NOTE

When programming Life Cycle, ensure that programming is not interrupted and can complete without error.

62.5 Initialization and application information

62.5.1 Reset

The reset state of each individual bit appears in the detailed register descriptions.

62.6 Additional safety measures

62.6.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM, but not other modules) by gating the state machines status with the expected status of the MC_RGM. If a spurious reset occurs, the SSCM will not interfere with the flash memory bus or overwrite configuration registers. However, internal SSCM status will be lost.

Chapter 63
Power Management Controller digital interface (PMC_dig)

63.1 Introduction

The power management controller (PMC) consists of two blocks: an analog block and a digital block. This chapter describes the digital block of the PMC.

The PMC_dig block contains all of the registers and digital logic to generate all of the enable signals, trim control bits, POR reset generation, and test mode logic for the analog block. The PMC digital logic also contains the controls for PMC analog outputs to be sensed with the ADC module.

In addition, the PMC digital logic contains the enable signals, trim bits, and other registers for the temperature sensor.

There are also several custom interfaces to various other blocks: The Reset Generation Module (MC_RGM) block receives the LVD/HVD values during POR to use for phase transition conditions.

There is also a custom interface to flash memory via the SSCM (and DCF client blocks) that is used to load the trim values during initial POR.

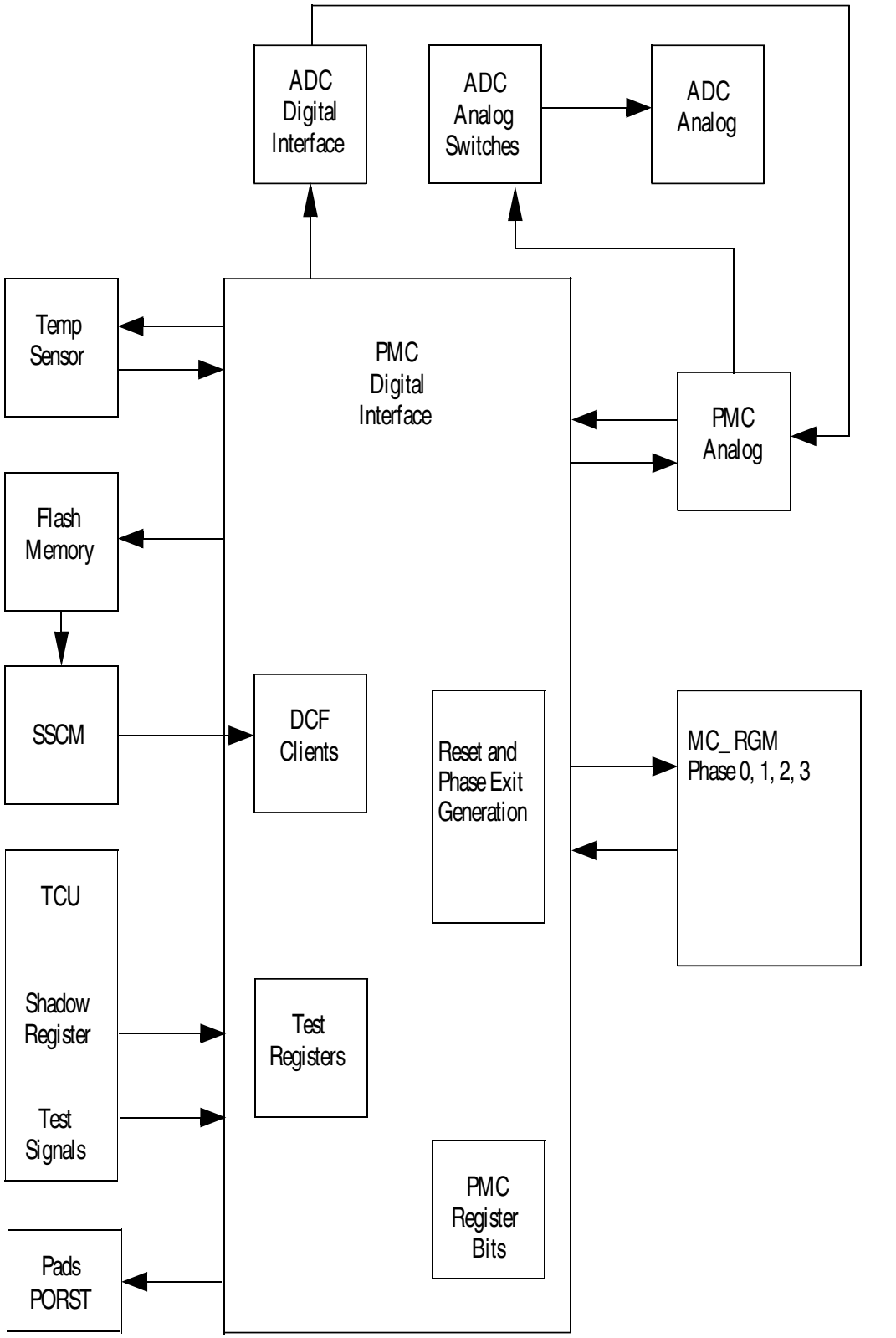
The TESTPIN must never be asserted to assure that the LVD/HVDs are active in the system.

The PMC digital block generates a transfer error event to the system if the selected address is greater than the highest address in the PMC digital memory map. A transfer error is not generated if an unused address within the PMC digital memory space is accessed.

NOTE

It is not valid to leave any of the power domains measured by an LVD or HVD unpowered. They must always be connected in such a way as to be in a valid operating range to get the device out of POR.

The following figure shows the block diagram.


ADC
Digital
Interface
ADC
Analog
Switches
ADC
Analog
Temp
Sensor
PMC
Digital
Interface
PMC
Analog
Flash
Memory
SSCM
DCF
Clients
Reset and
Phase Exit
Generation
MC_ RGM
Phase 0, 1, 2, 3
TCU
Shadow
Register
Test
Signals
Test
Registers
PMC
Register
Bits
Pads
PORST

63.2 Memory Map and Registers

The following table shows the PMC memory map. The PMC includes many registers for configuring, monitoring, and trimming the LVD monitors .

PMCDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Supply Gauge Status Register (PMCDIG_GR_S)	32	R	0000_0000h	63.2.1/3341
4	Pending Gauge Status Register (PMCDIG_GR_P)	32	R	0000_0000h	63.2.2/3344
8	Interrupt Enable Pending Register (PMCDIG_IE_P)	32	R/W	0000_0000h	63.2.3/3349
30	Event Pending Register (PMCDIG_EPR_VD3)	32	w1c	0000_0000h	63.2.4/3353
34	Reset Event Enable Register (PMCDIG_REE_VD3)	32	R/W	0000_00C1h	63.2.5/3355
38	Reset Event Select Register (PMCDIG_RES_VD3)	32	R/W	0000_0000h	63.2.6/3356
3C	FCCU Event Enable Register (PMCDIG_FEE_VD3)	32	R/W	0000_00C1h	63.2.7/3357
40	LVD108 Event Pending Register (PMCDIG_EPR_VD4)	32	w1c	0000_0000h	63.2.8/3359
44	Reset Event Select Register (PMCDIG_REE_VD4)	32	R/W	0000_0001h	63.2.9/3360
48	Reset Event Select Register (PMCDIG_RES_VD4)	32	R/W	0000_0000h	63.2.10/ 3361
4C	FCCU Event Enable Register (PMCDIG_FEE_VD4)	32	R/W	0000_0001h	63.2.11/ 3362
70	Event Pending Register (PMCDIG_EPR_VD7)	32	w1c	0000_0000h	63.2.12/ 3363
74	Reset Event Enable VD7 Register (PMCDIG_REE_VD7)	32	R/W	0000_0041h	63.2.13/ 3364
78	Reset Event Select Register (PMCDIG_RES_VD7)	32	R/W	0000_0000h	63.2.14/ 3365
7C	FCCU Event Enable Register (PMCDIG_FEE_VD7)	32	R/W	0000_0001h	63.2.15/ 3366
80	Event Pending Register (PMCDIG_EPR_VD8)	32	w1c	0000_0000h	63.2.16/ 3367
84	Reset Event Enable Register (PMCDIG_REE_VD8)	32	R/W	0000_0041h	63.2.17/ 3369
88	Reset Event Select Registerr (PMCDIG_RES_VD8)	32	R/W	0000_0000h	63.2.18/ 3370
8C	FCCU Event Enable Register (PMCDIG_FEE_VD8)	32	R/W	0000_0041h	63.2.19/ 3371
90	LVD270 Event Pending Register (PMCDIG_EPR_VD9)	32	w1c	0000_0000h	63.2.20/ 3372
94	Reset Event Enable Register (PMCDIG_REE_VD9)	32	R/W	0000_4341h	63.2.21/ 3375

Table continues on the next page...

PMCDIG memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
98	Reset Event Select Register (PMCDIG_RES_VD9)	32	R/W	0000_0000h	63.2.22/ 3377
9C	FCCU Event Enable VD9 (PMCDIG_FEE_VD9)	32	R/W	0000_4341h	63.2.23/ 3379
A0	LVD295 Event Pending Register (PMCDIG_EPR_VD10)	32	w1c	0000_0000h	63.2.24/ 3381
A4	Reset Event Enable Register (PMCDIG_REE_VD10)	32	R/W	0000_8040h	63.2.25/ 3383
AC	FCCU Event Enable Register (PMCDIG_FEE_VD10)	32	R/W	0000_8040h	63.2.26/ 3384
C0	HVD360 Event Pending Register (PMCDIG_EPR_VD12)	32	w1c	0000_0000h	63.2.27/ 3385
C4	Reset Event Enable Register (PMCDIG_REE_VD12)	32	R/W	0000_0040h	63.2.28/ 3386
C8	Reset Event Select Register (PMCDIG_RES_VD12)	32	R/W	0000_0000h	63.2.29/ 3387
CC	FCCU Event Enable Register (PMCDIG_FEE_VD12)	32	R/W	0000_0040h	63.2.30/ 3388
D0	Event Pending Register (PMCDIG_EPR_VD13)	32	w1c	0000_0000h	63.2.31/ 3389
D4	Reset Event Enable Register (PMCDIG_REE_VD13)	32	R/W	0000_0100h	63.2.32/ 3390
D8	Reset Event Select Register (PMCDIG_RES_VD13)	32	R/W	0000_0000h	63.2.33/ 3391
DC	FCCU Event Enable Register (PMCDIG_FEE_VD13)	32	R/W	0000_0100h	63.2.34/ 3392
E0	Event Pending Register (PMCDIG_EPR_VD14)	32	w1c	0000_0000h	63.2.35/ 3393
E4	Reset Event Enable Register (PMCDIG_REE_VD14)	32	R/W	0000_8100h	63.2.36/ 3395
E8	Reset Event Select Register (PMCDIG_RES_VD14)	32	R/W	0000_0000h	63.2.37/ 3396
EC	FCCU Event Enable Register (PMCDIG_FEE_VD14)	32	R/W	0000_8100h	63.2.38/ 3397
F0	Event Pending Register (PMCDIG_EPR_VD15)	32	w1c	0000_0000h	63.2.39/ 3398
F4	Reset Event Enable Register (PMCDIG_REE_VD15)	32	R/W	0000_8001h	63.2.40/ 3400
F8	Reset Event Select Register (PMCDIG_RES_VD15)	32	R/W	0000_0000h	63.2.41/ 3401
FC	FCCU Event Enable Register (PMCDIG_FEE_VD15)	32	R/W	0000_8001h	63.2.42/ 3402
104	Voltage Supply for I/O Segment Register (PMCDIG_VSIO)	32	R/W	0000_0F00h	63.2.43/ 3403

Table continues on the next page...

PMCDIG memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
300	Event Pending Register (PMCDIG_EPR_TD)	32	w1c	0000_0000h	63.2.44/ 3404
304	Reset Event Enable Register (PMCDIG_REE_TD)	32	R/W	0000_0000h	63.2.45/ 3405
308	Reset Event Select Register (PMCDIG_RES_TD)	32	R/W	0000_0000h	63.2.46/ 3406
30C	Temperature Sensor Configuration Register (PMCDIG_CTL_TD)	32	R/W	0000_0003h	63.2.47/ 3408
318	Temp Sensor FCCU Event Enable Register (PMCDIG_FEE_TD)	32	R/W	0000_000Dh	63.2.48/ 3410
340	Voltage Detect User Mode Test Register (PMCDIG_VD_UTST)	32	R/W	0F00_0000h	63.2.49/ 3411
344	ADC Channel Select Register (PMCDIG_ADC_CH)	32	R/W	0000_0000h	63.2.50/ 3413
34C	Voltage Regulator 1.2V Control Register (PMCDIG_VREG1P2_CTRL)	32	R/W	0000_0000h	63.2.51/ 3413
350	Module Control Register (PMCDIG_MCR)	32	w1c	0000_0000h	63.2.52/ 3414

63.2.1 Supply Gauge Status Register (PMCDIG_GR_S)

This status register contains the gauge of the indicated supply (or temperature sensor value). This indicates the present state of the voltage (temperature) detect events for each of the supplies. Each of the voltage detect signals for a given voltage are combined to form a single bit.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					TS3	TS2	TS0	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD15	VD14	VD13	VD12	0	VD10	VD9	VD8	VD7	0		VD4	VD3	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_GR_S field descriptions

Field	Description
0–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5 TS3	TS3 over temperature detect flag. This read-only bit is the over temperature status flag associated with the high temp sensor point (165C). It is asserted when the temperature exceeds its corresponding threshold, and clears when all the temperature falls below this threshold. 0 Currently no occurrence. 1 Temp Sensor is currently above the hot temperature threshold.

Table continues on the next page...

PMCDIG_GR_S field descriptions (continued)

Field	Description
6 TS2	TS2 over temperature detect flag. This read-only bit is the over temperature status flag associated with the high temp sensor point (150C). It is asserted when the temperature exceeds its corresponding threshold, and clears when all the temperature falls below this threshold. 0 Currently no occurrence. 1 Temp Sensor is currently above the temp threshold.
7 TS0	TS0 under temperature detect flag. This read-only bit is the undertemperature status flag associated with the cold temp sensor point. It is asserted when the temperature goes below its corresponding threshold, and clears when all the temperature rises above this threshold. 0 Currently no occurrence. 1 Temp Sensor is currently below the cold temp threshold.
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 VD15	VD15 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 6.00 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 6.00 V supply.
17 VD14	VD14 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold. All the 4.00 V supplies are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 4.00 V supplies.
18 VD13	VD13 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3.6 V supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold.All the 3.6 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage 3.6V supply.
19 VD12	VD12 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 6.00 V HVDs are combined to form this bit.

Table continues on the next page...

PMCDIG_GR_S field descriptions (continued)

Field	Description
	0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 3.60 V supply.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 VD10	HVD occurrence detected on the high voltage 3.60 V supply. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 3.00 V supply.
22 VD9	VD9 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold. All of the 2.8 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.8 V supplies.
23 VD8	VD8 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 1.45 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 1.45 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage 1.45 V supplies.
24 VD7	VD7 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 1.40 V supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All the 1.40 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage 1.40 V supplies.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 VD4	VD4 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detects for the low voltage 1.14 V supply associated with the PLL, Flash, and hot point. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage 1.14 V point supply.
28 VD3	VD3 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V supply associated with the core logic. It is asserted when any of the VD3 supplies fall below the

Table continues on the next page...

PMCDIG_GR_S field descriptions (continued)

Field	Description
	corresponding LVD threshold, and clears when the supplies rise above the corresponding LVD threshold. All the 1.14 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on any of the low voltage 1.14 V supplies.
29–31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

63.2.2 Pending Gauge Status Register (PMCDIG_GR_P)

This configuration register contains a mirror of the contents of the Event Pending Registers, but in a format that includes one bit for each voltage detect signal. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the EPR register). These bits are read only, and they are cleared by writing a one to the corresponding EPR register.

The Pending registers are only set on an active edge of an event, so a transition must be seen before they will become set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						VD15_A	VD15_C	0	VD14_A	VD14_IM	0		VD13_IM	VD12_F	VD10_A
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD10_F	VD9_O	VD9_IF2	VD9_IF	VD9_IJ	VD9_IM	VD9_F	VD9_EBI	VD9_C	VD8_C	VD8_F	VD7_C	VD4_C	VD3_P	VD3_F	VD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_GR_P field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 VD15_A	VD15_A high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V ADC supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD15[HVD15_A] is cleared. 0 No occurrence has happened, or EPR_VD15[HVD15_A] has been cleared. 1 HVD occurrence detected on the high voltage ADC supply.
7 VD15_C	VD15_C high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD15[HVD15_C] is cleared. 0 No occurrence has happened, or EPR_VD15[HVD15_C] has been cleared. 1 HVD occurrence detected on the high voltage cold point supply.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 VD14_A	VD14_A low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD14[LVD14_A] is cleared. 0 No occurrence has happened, or EPR_VD14[LVD14_A] has been cleared. 1 LVD occurrence detected on the high voltage ADC supply.
10 VD14_IM	VD14_IM low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD14[LVD14_IM] is cleared.

Table continues on the next page...

PMCDIG_GR_P field descriptions (continued)

Field	Description
	0 No occurrence has happened, or EPR_VD14[LVD14_IM] has been cleared. 1 LVD occurrence detected on the high voltage I/O main supply.
11–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 VD13_IM	VD13_IM low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3.6 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD13[LVD13_IM] is cleared. 0 No occurrence has happened, or EPR_VD13[LVD13_IM] has been cleared. 1 LVD occurrence detected on the high voltage I/O main supply.
14 VD12_F	VD12_F high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage 3.6 V flash supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD12[HVD12_F] is cleared. 0 No occurrence has happened, or EPR_VD12[HVD12_F] has been cleared. 1 HVD occurrence detected on the high voltage flash supply.
15 VD10_A	VD10_A low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD10[LVD10_A] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD10[LVD10_A] = 1. 0 No occurrence has happened, or EPR_VD10[LVD10_A] has been cleared. 1 LVD occurrence detected on the high voltage ADC supply.
16 VD10_F	VD10_F low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD10[LVD10_F] is cleared. 0 No occurrence has happened, or EPR_VD10[LVD10_F] has been cleared. 1 LVD occurrence detected on the high voltage flash supply.
17 VD9_O	VD9_O low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V OSC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_O] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_O] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_O] has been cleared. 1 LVD occurrence detected on the high voltage OSC supply.
18 VD9_IF2	VD9_IF2 low-voltage detect flag.

Table continues on the next page...

PMCDIG_GR_P field descriptions (continued)

Field	Description
	This read-only bit is the low-voltage status flag associated with the voltage level detect for the second high voltage 2.8 V I/O Flexray supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IF2] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_IF] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_IF2] has been cleared. 1 LVD occurrence detected on the second high voltage I/O Flexray supply.
19 VD9_IF	VD9_IF low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O Flexray supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IF] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_IF] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_IF] has been cleared. 1 LVD occurrence detected on the high voltage I/O Flexray supply.
20 VD9_IJ	VD9_IJ low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O JTAG supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IJ] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_IJ] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_IJ] has been cleared. 1 LVD occurrence detected on the high voltage I/O JTAG supply.
21 VD9_IM	VD9_IM low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_IM] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_IM] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_IM] has been cleared. 1 LVD occurrence detected on the high voltage I/O main supply.
22 VD9_F	VD9_F low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_F] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_F] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_F] has been cleared. 1 LVD occurrence detected on the high voltage flash supply.
23 VD9_EBI	VD9_EBI low-voltage detect flag.

Table continues on the next page...

PMCDIG_GR_P field descriptions (continued)

Field	Description
	This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O External Bus Interface (EBI) supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_EBI] is cleared. 0 No occurrence has happened, or EPR_VD9[LVD9_EBI] has been cleared. 1 LVD occurrence detected on the high voltage I/O External Bus Interface supply.
24 VD9_C	VD9_C low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD9[LVD9_C] is cleared. This bit is indeterminate after reset, so it must be cleared during initialization by writing PMCDIG_EPR_VD9[LVD9_C] = 1. 0 No occurrence has happened, or EPR_VD9[LVD9_C has been cleared. 1 LVD occurrence detected on the high voltage cold point supply.
25 VD8_C	VD8_C high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.45 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD8[HVD8_C] is cleared. 0 No occurrence has happened, or EPR_VD8[HVD8_C] has been cleared. 1 LVD occurrence detected on the low voltage cold point supply.
26 VD8_F	VD8_F high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.45 V flash supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD8[HVD8_F] is cleared. 0 No occurrence has happened, or EPR_VD7[HVD7_F] has been cleared. 1 HVD occurrence detected on the low voltage flash supply.
27 VD7_C	VD7_C high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.4 V cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears when the EPR_VD7[HVD7_C] is cleared. 0 No occurrence has happened, or EPR_VD7[HVD7_C] has been cleared. 1 HVD occurrence detected on the low voltage cold point supply.
28 VD4_C	VD4_C low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD4[LVD4_C] is cleared. 0 No occurrence has happened, or EPR_VD4[LVD4_C] has been cleared. 1 LVD occurrence detected on the low voltage cold point supply.
29 VD3_P	VD3_P low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V PLL supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD3[LVD3_P] is cleared.

Table continues on the next page...

PMCDIG_GR_P field descriptions (continued)

Field	Description
	0 No occurrence has happened, or EPR_VD3[LVD3_P] has been cleared. 1 LVD occurrence detected on the low voltage PLL supply.
30 VD3_F	VD3_F low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD3[LVD3_F] is cleared. 0 No occurrence has happened, or EPR_VD3[LVD3_F] has been cleared. 1 LVD occurrence detected on the low voltage flash supply.
31 VD3_C	VD3_C low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V hot point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the EPR_VD3[LVD3_C] is cleared. 0 No occurrence has happened, or EPR_VD3[LVD3_C] has been cleared. 1 LVD occurrence detected on the low voltage hot point supply.

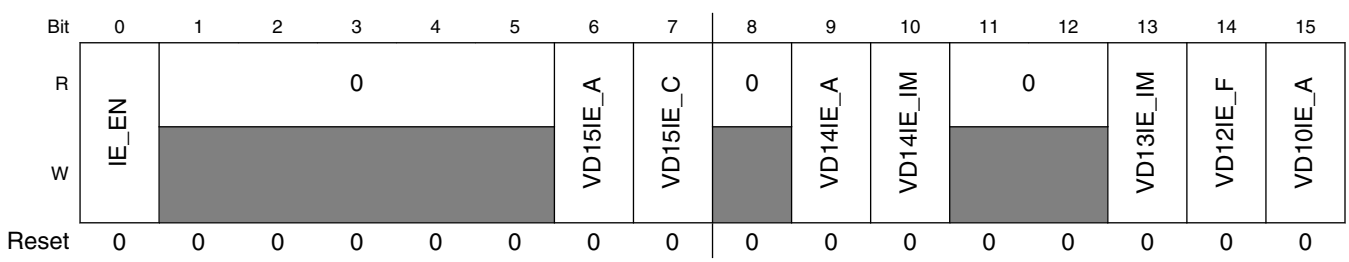
63.2.3 Interrupt Enable Pending Register (PMCDIG_IE_P)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending Registers (EPR_XX), but in a format similar to the Pending Gauge Status register (GR_P). These bits indicate whether an interrupt occurs when the voltage event is seen. A ‘0’ indicates that no interrupt is to be sent, and a ‘1’ indicates that an interrupt is to occur when the voltage detect event occurs. These bits are enabled via a write of one to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

NOTE

Special care must be taken with the EPR_VD9, EPR_VD10 and the interrupt enables of the IE_P reguster. EPR_VD9 and EPR_VD10 must be cleared before enabling the interrupts for VD9IE and VD10IE voltage detection.

Address: 0h base + 8h offset = 8h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	VD10IE_F	VD9IE_O	VD9IE_IF2	VD9IE_IF	VD9IE_IJ	VD9IE_IM	VD9IE_F	VD9IE_EBI	VD9IE_C	VD8IE_C	VD8IE_F	VD7IE_C	VD4IE_C	VD3IE_P	VD3IE_F	VD3IE_C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_IE_P field descriptions

Field	Description
0 IE_EN	IE_EN. Interrupt Enable. This bit determines whether any of the Interrupt Enable bits can be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 VD15IE_A	VD15IE_A Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
7 VD15IE_C	VD15IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 VD14IE_A	VD14IE_A Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
10 VD14IE_IM	VD14IE_IM Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
11–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 VD13IE_IM	VD13IE_IM Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
14 VD12IE_F	VD12IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

Table continues on the next page...

PMCDIG_IE_P field descriptions (continued)

Field	Description
15 VD10IE_A	VD10IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
16 VD10IE_F	VD10IE_A Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
17 VD9IE_O	VD9IE_O Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
18 VD9IE_IF2	VD9IE_IF2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
19 VD9IE_IF	VD9IE_IF Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
20 VD9IE_IJ	VD9IE_IJ Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
21 VD9IE_IM	VD9IE_IM Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
22 VD9IE_F	VD9IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
23 VD9IE_EBI	VD9IE_EBI Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
24 VD9IE_C	VD9IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

Table continues on the next page...

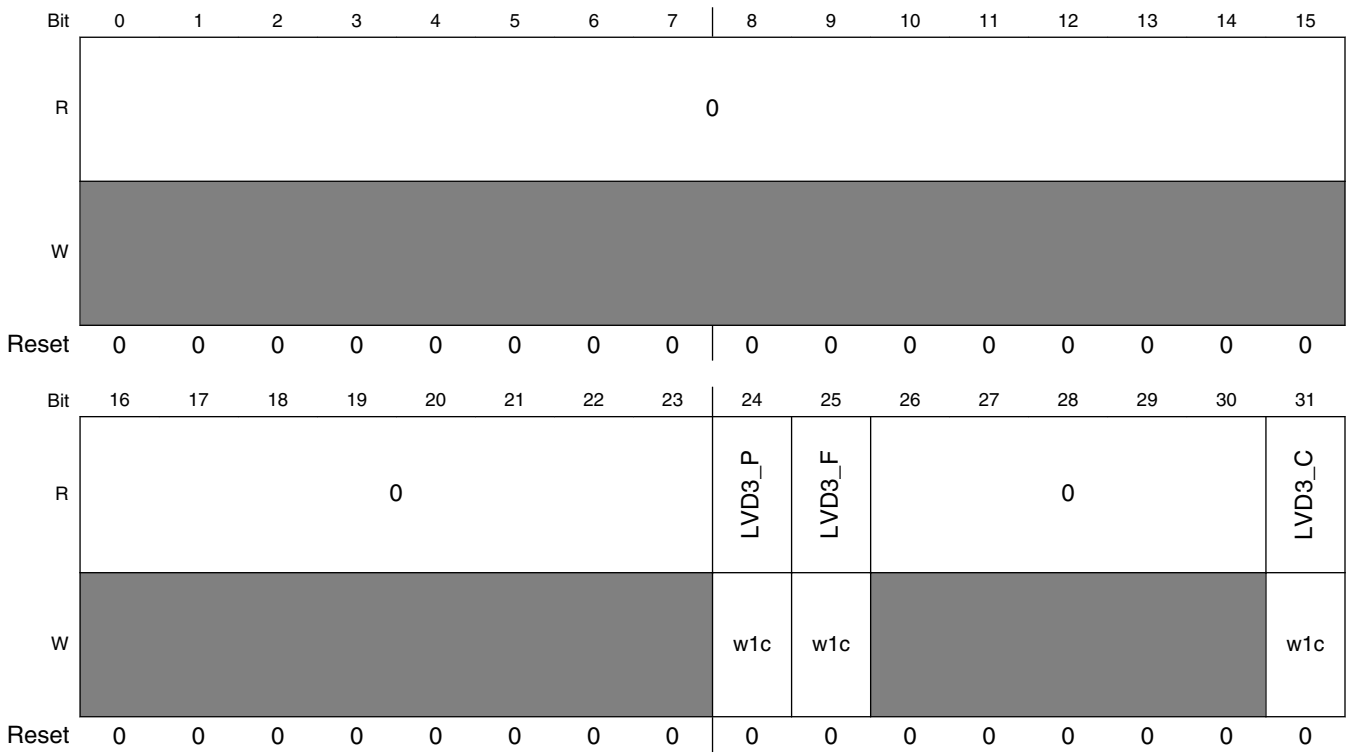
PMCDIG_IE_P field descriptions (continued)

Field	Description
25 VD8IE_C	VD8IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
26 VD8IE_F	VD8IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
27 VD7IE_C	VD7IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
28 VD4IE_C	VD4IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. VD3IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
29 VD3IE_P	VD3IE_P Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
30 VD3IE_F	VD3IE_F Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
31 VD3IE_C	VD3IE_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

63.2.4 Event Pending Register (PMCDIG_EPR_VD3)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. To clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + 30h offset = 30h



PMCDIG_EPR_VD3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LVD3_P	LVD3_P flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.08 V PLL supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD3IE_P bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD3[LVD3_P] is also asserted, a system reset is generated, which clears VD3IE_P and negates the interrupt request. If REE_VD3[LVD3_P] is asserted and a destructive reset is selected (via RES_VD3[LVD3_P] being 0), then a destructive reset is generated. If RES_VD3[LVD3_P] is set, then a functional reset is generated.

Table continues on the next page...

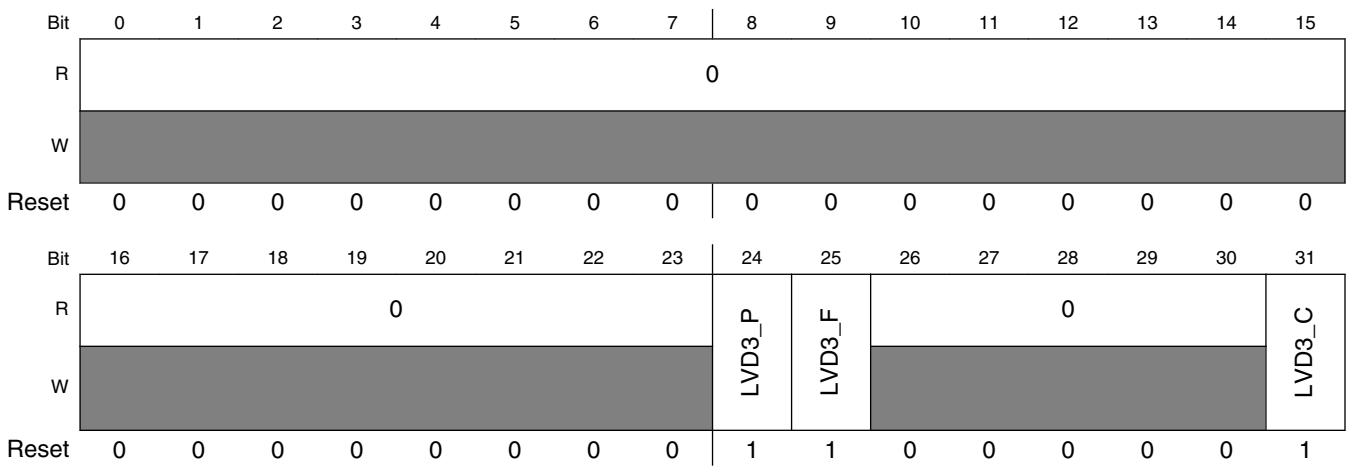
PMCDIG_EPR_VD3 field descriptions (continued)

Field	Description
	0 Currently no occurrence. 1 LVD occurrence detected on the low voltage PLL supply.
25 LVD3_F	LVD3_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.08 V flash supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD3IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD3[LVD3_F] is also asserted, a system reset is generated, which clears VD3IE_F and negatse the interrupt request. If REE_VD3[LVD3_F] is asserted and a destructive reset is selected (via RES_VD3[LVD3_F] being 0), then a destructive reset is generated. If RES_VD3[LVD3_F] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage flash supply.
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD3_C	LVD3_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.08 V hot point supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD3IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD3[LVD3_C] is also asserted, a system reset is generated, which clears VD3IE_C and negates the interrupt request. If REE_VD3[LVD3_C] is asserted and a destructive reset is selected (via RES_VD3[LVD3_C] being 0), then a destructive reset is generated. If RES_VD3[LVD3_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage 1.14 V hot point supply.

63.2.5 Reset Event Enable Register (PMCDIG_REE_VD3)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + 34h offset = 34h



PMCDIG_REE_VD3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LVD3_P	LVD3_P reset enable. This bit defines whether an LVD assertion on the supply of the PLL generates a system reset. The RES_VD3[LVD3_P] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the PLL does not cause system reset. 1 Enabled. LVD assertion on the supply of the PLL causes system reset.
25 LVD3_F	LVD3_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. The RES_VD3[LVD3_F] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD3_C	LVD3_C reset enable.

Table continues on the next page...

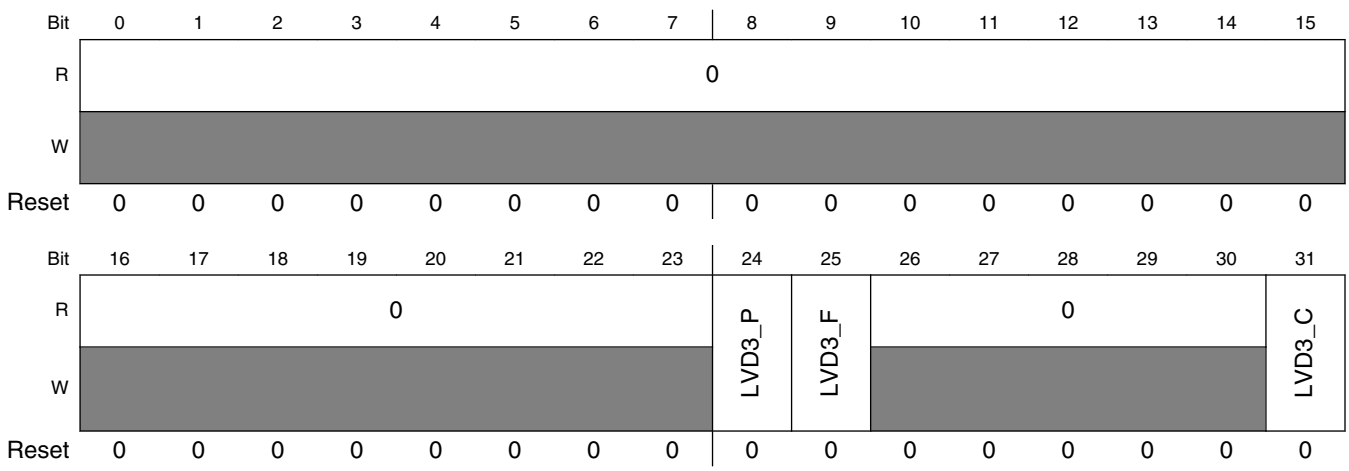
PMCDIG_REE_VD3 field descriptions (continued)

Field	Description
	This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD3[LVD3_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

63.2.6 Reset Event Select Register (PMCDIG_RES_VD3)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + 38h offset = 38h



PMCDIG_RES_VD3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LVD3_P	LVD3_P reset event select. This bit defines whether an LVD assertion on the supply of the PLL generates a destructive reset or a functional reset. The REE_VD3[LVD3_P] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the PLL causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the PLL causes a functional system reset.

Table continues on the next page...

PMCDIG_RES_VD3 field descriptions (continued)

Field	Description
25 LVD3_F	LVD3_F reset event select. This bit defines whether an LVD assertion on the supply of the flash generates a destructive reset or a functional reset. The REE_VD3[LVD3_F] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the flash causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the flash causes a functional system reset.
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD3_C	LVD3_C reset event select. This bit defines whether an LVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD3[LVD3_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

63.2.7 FCCU Event Enable Register (PMCDIG_FEE_VD3)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								FEE3_P	FEE3_F	0					FEE3_C
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1

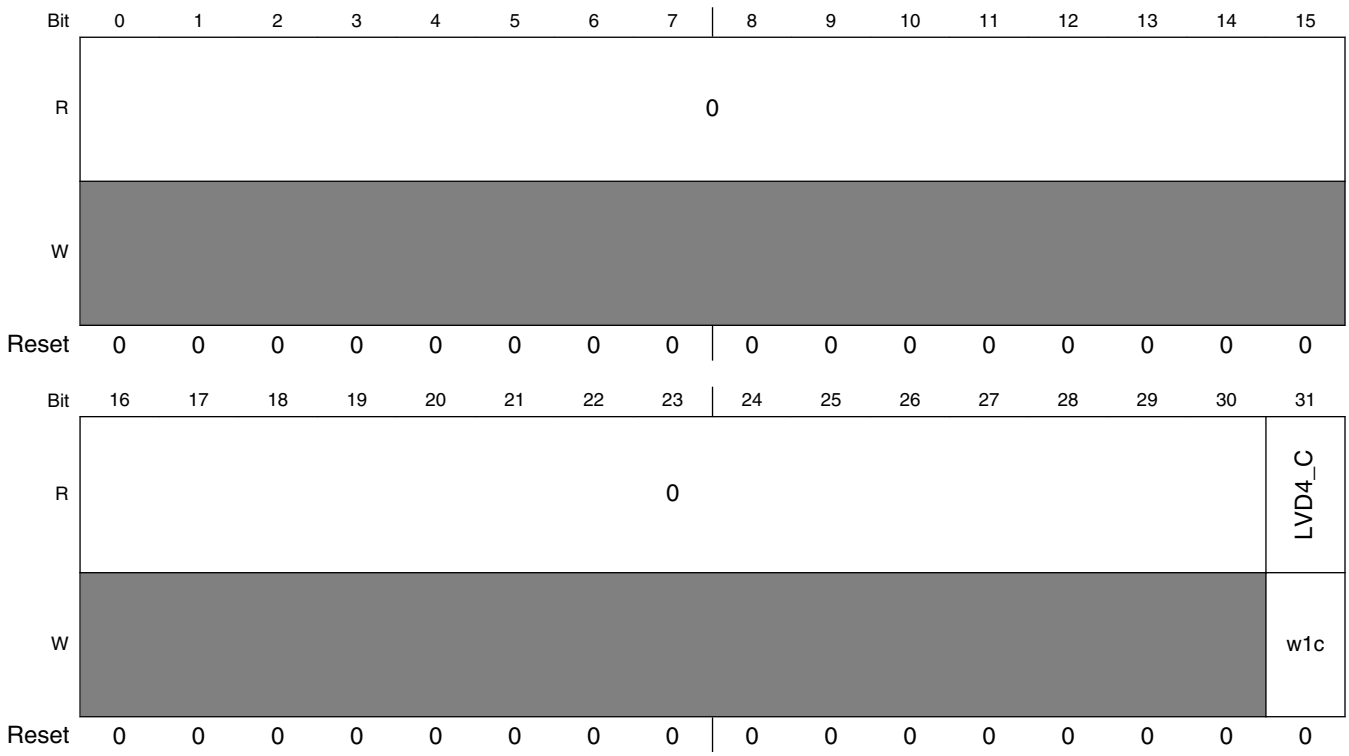
PMCDIG_FEE_VD3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 FEE3_P	FEE3_P FCCU event enable. This bit defines whether an LVD assertion on the supply of the PLL generates an FCCU event. 0 Disabled. LVD assertion on the supply of the PLL does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the PLL causes an FCCU event.
25 FEE3_F	FEE3_F FCCU event enable. This bit defines whether an LVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. LVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the flash causes an FCCU event.
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE3_C	FEE3_C FCCU event enable. This bit defines whether an LVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. LVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the cold point causes an FCCU event.

63.2.8 LVD108 Event Pending Register (PMCDIG_EPR_VD4)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + 40h offset = 40h



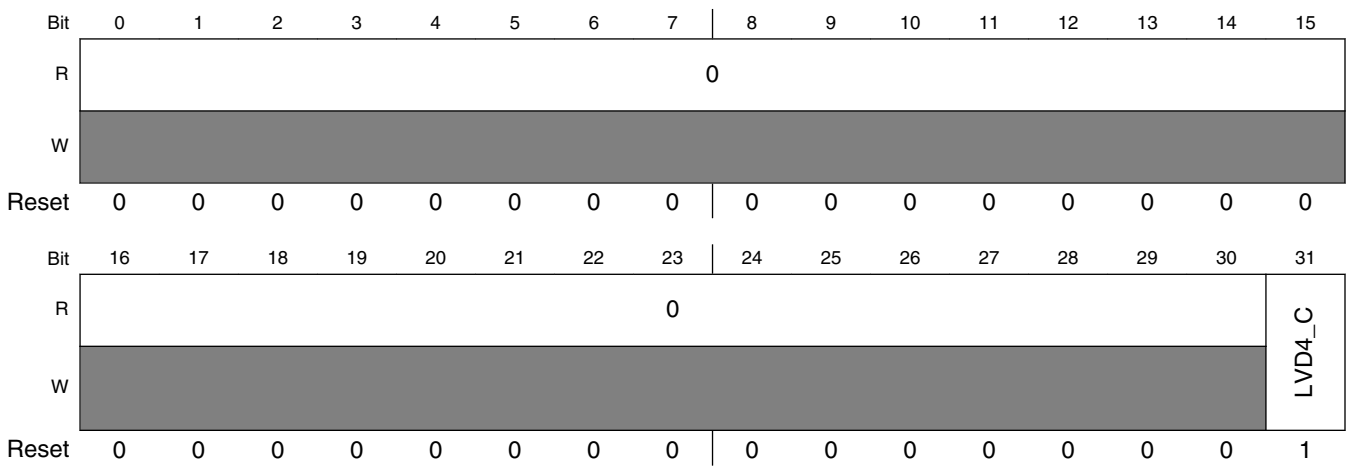
PMCDIG_EPR_VD4 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD4_C	LVD4_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the low voltage 1.14 V cold point supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD4IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD4[LVD4_C] is also asserted, a system reset is generated, which clears VD4IE_C and negates the interrupt request. If REE_VD4[LVD4_C] is asserted and a destructive reset is selected (via RES_VD4[LVD4_C] being 0), then a destructive reset is generated. If RES_VD4[LVD4_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage cold point supply.

63.2.9 Reset Event Select Register (PMCDIG_REE_VD4)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + 44h offset = 44h



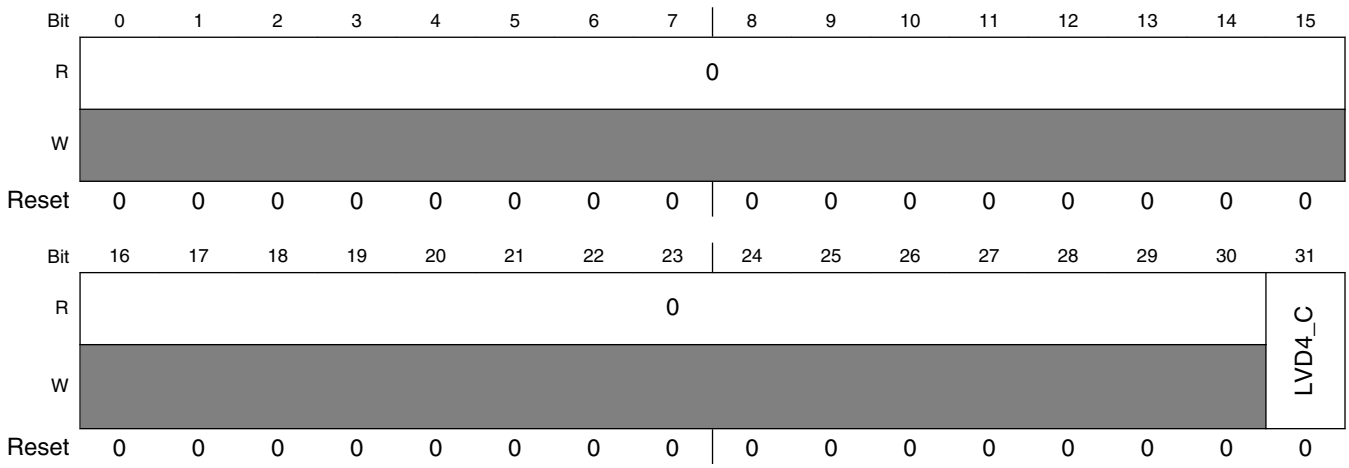
PMCDIG_REE_VD4 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD4_C	LVD4_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a system reset. The RES_VD4[LVD4_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

63.2.10 Reset Event Select Register (PMCDIG_RES_VD4)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + 48h offset = 48h



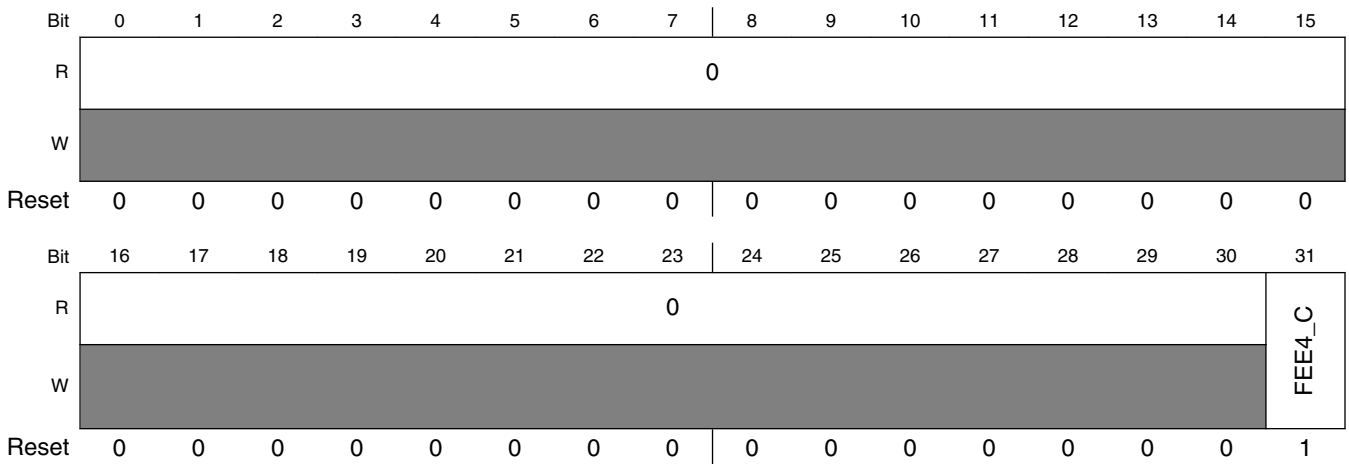
PMCDIG_RES_VD4 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD4_C	LVD4_H reset event select. This bit defines whether an LVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD4[LVD4_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

63.2.11 FCCU Event Enable Register (PMCDIG_FEE_VD4)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + 4Ch offset = 4Ch



PMCDIG_FEE_VD4 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE4_C	FEE4_C FCCU event enable. This bit defines whether an LVD assertion on the supply of the cold point generates an FCCU event.. 0 Disabled. LVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the cold point causes an FCCU event.

63.2.12 Event Pending Register (PMCDIG_EPR_VD7)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									0	0					HVD7_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_EPR_VD7 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD7_C	HVD7_C flag. This bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.4 V cold point supply. It is asserted asynchronously when the supply rises above its corresponding HVD

Table continues on the next page...

PMCDIG_EPR_VD7 field descriptions (continued)

Field	Description
	threshold, and clears when a one is written to this bit location. If the VD7IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD7[HVD7_C] is also asserted, a system reset is generated, which clears VD7IE_C and negates the interrupt request. If REE_VD7[HVD7_C] is asserted and a destructive reset is selected (via RES_VD7[HVD7_C] being 0), then a destructive reset is generated. If RES_VD7[HVD7_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage cold point supply.

63.2.13 Reset Event Enable VD7 Register (PMCDIG_REE_VD7)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									0	0					HVD7_C
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

PMCDIG_REE_VD7 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

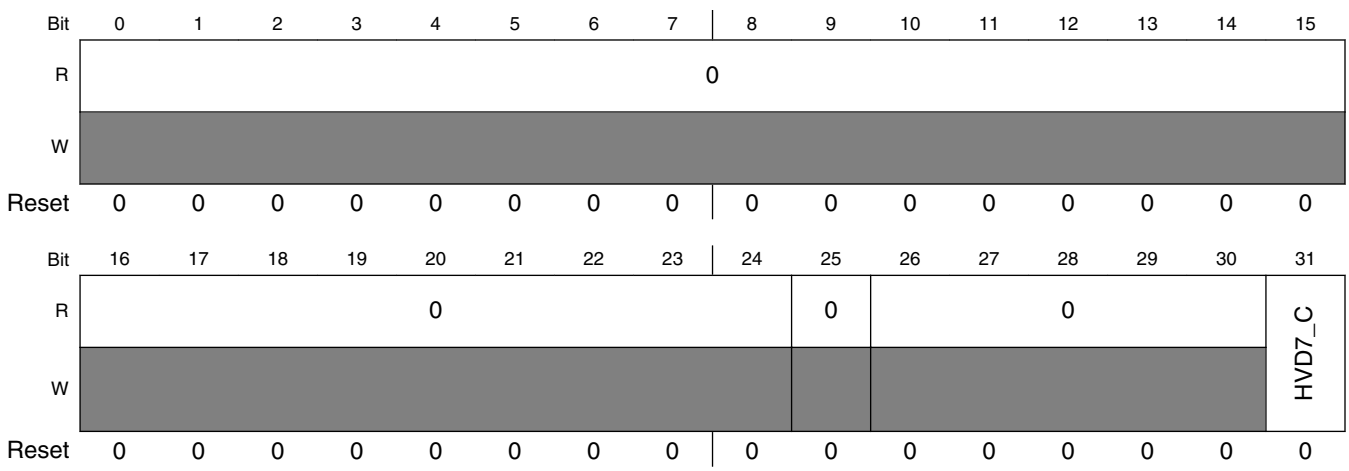
PMCDIG_REE_VD7 field descriptions (continued)

Field	Description
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD7_C	HVD7_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD7[HVD7_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

63.2.14 Reset Event Select Register (PMCDIG_RES_VD7)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + 78h offset = 78h



PMCDIG_RES_VD7 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

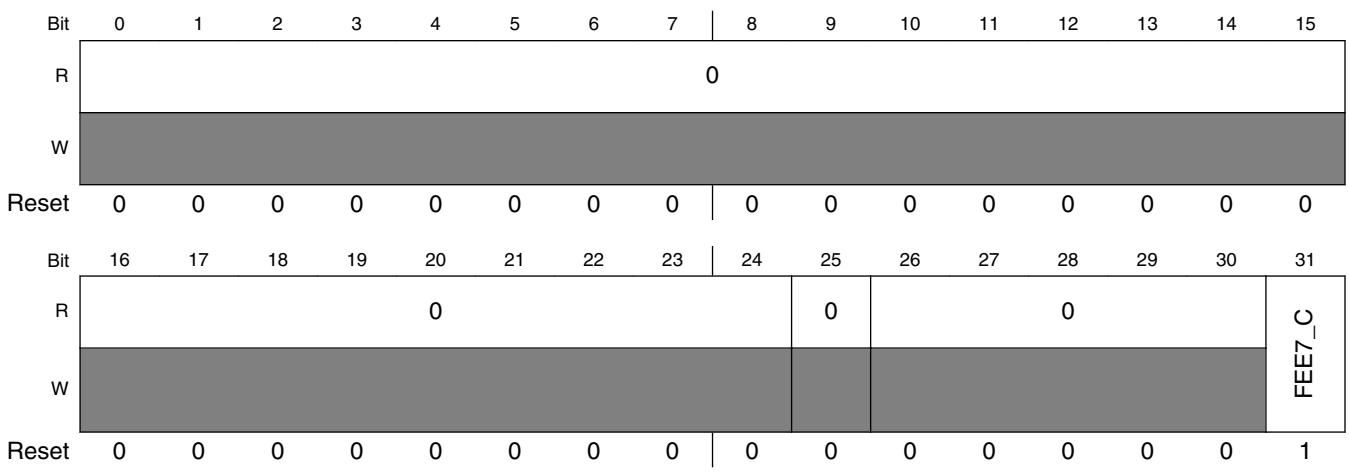
PMCDIG_RES_VD7 field descriptions (continued)

Field	Description
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD7_C	HVD7_C reset event select. This bit defines whether an HVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD7[HVD7_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

63.2.15 FCCU Event Enable Register (PMCDIG_FEE_VD7)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + 7Ch offset = 7Ch



PMCDIG_FEE_VD7 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMCDIG_FEE_VD7 field descriptions (continued)

Field	Description
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE7_C	FEE7_C FCCU event enable. This bit defines whether an HVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. HVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the cold point causes an FCCU event.

63.2.16 Event Pending Register (PMCDIG_EPR_VD8)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									HVD8_F	0					HVD8_C
W										w1c						w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

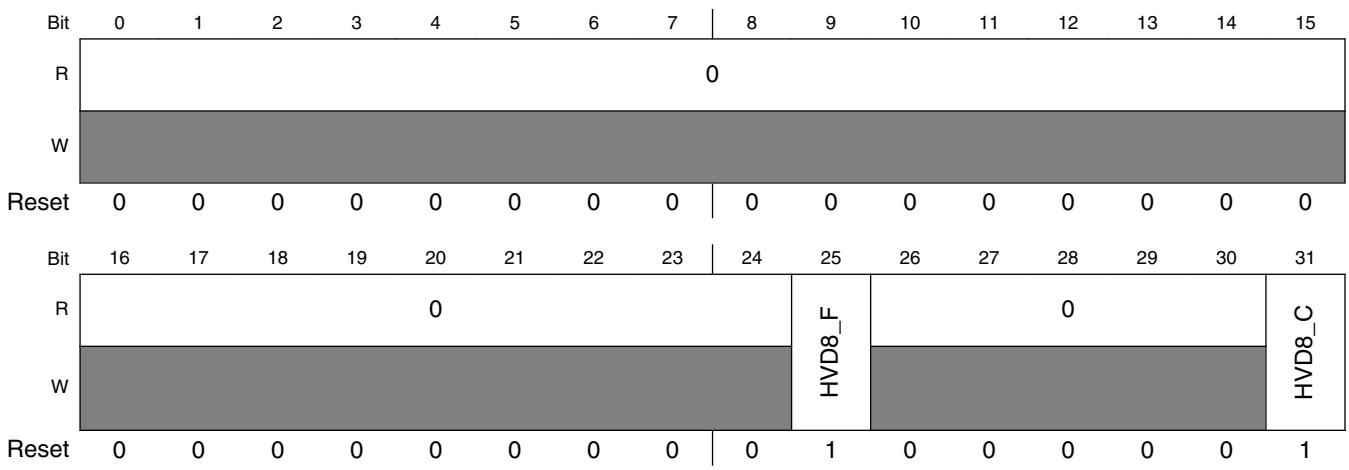
PMCDIG_EPR_VD8 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 HVD8_F	HVD8_F flag. This bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.45 V Flash supply. It is asserted asynchronously when the supply rises above its corresponding HVD threshold, and clears when a one is written to this bit location. If the VD8IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD8[HVD8_F] is also asserted, a system reset is generated, which clears VD8IE_F and negates the interrupt request. If REE_VD8[HVD8_F] is asserted and a destructive reset is selected (via RES_VD8[HVD8_F] being 0), then a destructive reset is generated. If RES_VD8[HVD8_F] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage Flash supply.
26–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD8_C	HVD8_C flag. This bit is the high-voltage status flag associated with the voltage level detect for the low voltage 1.45 V cold point supply. It is asserted asynchronously when the supply rises above its corresponding HVD threshold, and clears when a one is written to this bit location. If the VD8IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD8[HVD8_C] is also asserted, a system reset is generated, which clears VD8IE_C and negates the interrupt request. If REE_VD8[HVD8_C] is asserted and a destructive reset is selected (via RES_VD8[HVD8_C] being 0), then a destructive reset is generated. If RES_VD8[HVD8_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage cold point supply.

63.2.17 Reset Event Enable Register (PMCDIG_REE_VD8)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) unless in test mode.

Address: 0h base + 84h offset = 84h



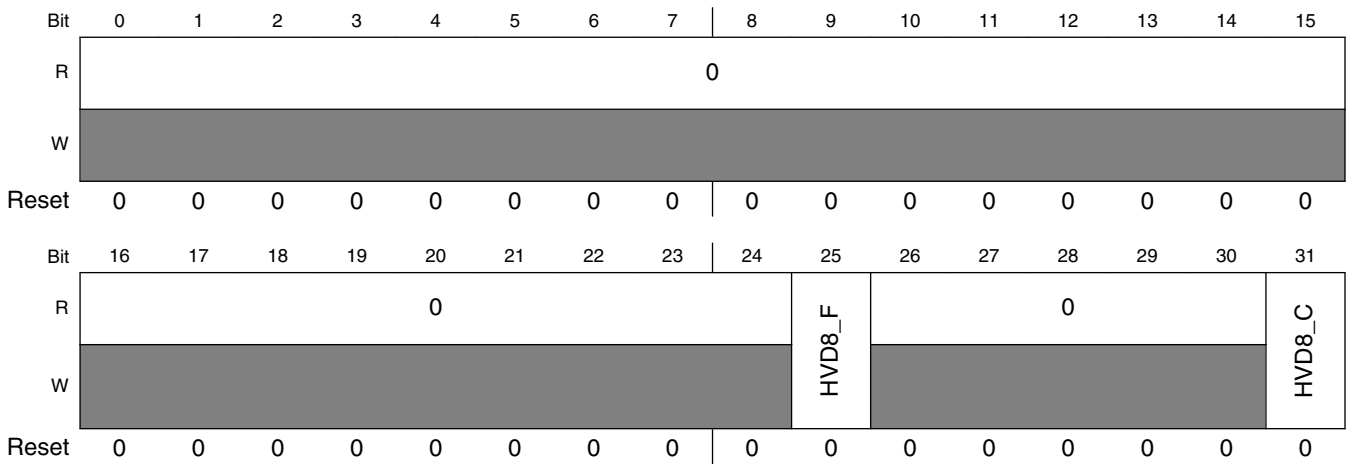
PMCDIG_REE_VD8 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 HVD8_F	HVD8_F reset enable. This bit defines whether an HVD assertion on the supply of the Flash generates a system reset. The RES_VD8[HVD8_F] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the Flash does not cause system reset. 1 Enabled. HVD assertion on the supply of the Flash causes system reset.
26–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD8_C	HVD8_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD8[HVD8_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

63.2.18 Reset Event Select Registerr (PMCDIG_RES_VD8)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + 88h offset = 88h



PMCDIG_RES_VD8 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 HVD8_F	HVD8_F reset event select This bit defines whether an HVD assertion on the supply of the Flash generates a destructive reset or a functional reset. The REE_VD8[HVD8_F] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the Flash causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the Flash causes a functional system reset.
26–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD8_C	HVD8_C reset event select. This bit defines whether an HVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD8[HVD8_C] bit determines whether the reset event is enabled.

Table continues on the next page...

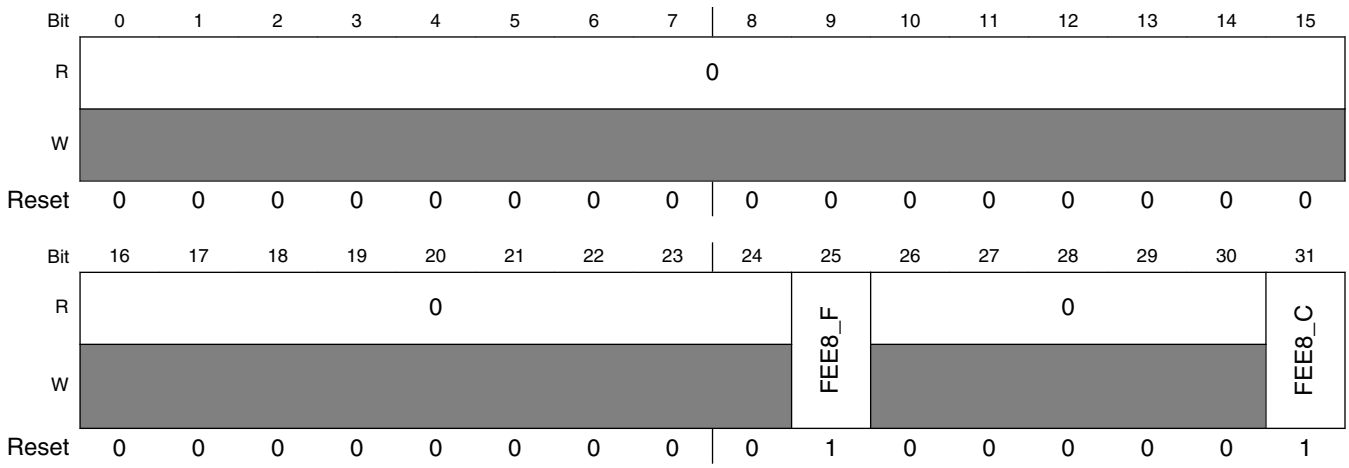
PMCDIG_RES_VD8 field descriptions (continued)

Field	Description
	0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

63.2.19 FCCU Event Enable Register (PMCDIG_FEE_VD8)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + 8Ch offset = 8Ch



PMCDIG_FEE_VD8 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FEE8_F	FEE8_F FCCU event enable. This bit defines whether an HVD assertion on the supply of the Flash generates an FCCU event. 0 Disabled. HVD assertion on the supply of the Flash does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the Flash causes an FCCU event.
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE8_C	FEE8_C FCCU event enable. This bit defines whether an HVD assertion on the supply of the cold point generates an FCCU event.

Table continues on the next page...

PMCDIG_FEE_VD8 field descriptions (continued)

Field	Description
	0 Disabled. HVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the cold point causes an FCCU event.

63.2.20 LVD270 Event Pending Register (PMCDIG_EPR_VD9)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

NOTE

These bits are cleared out of initial power up POR, but are not reset during other LVD, HVD, and POR events. They must be cleared by software during initialization by writing 1 to the EPR_VD9 field. The field values are retained until the core voltage is less than 0.8V).

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LVD9_O	0		LVD9_IF2	LVD9_IF	LVD9_IJ	LVD9_IM	0	LVD9_F	0		LVD9_EBI	0		LVD9_C
W		w1c			w1c	w1c	w1c	w1c		w1c			w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_EPR_VD9 field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 LVD9_O	LVD9_O flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V OSC supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. If the VD9IE_O bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVDn_O] is also asserted, a system reset is generated, which clears VD9IE_O and negates the interrupt request. If REE_VD9[LVD9_O] is asserted, a POR reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage OSC supply.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 LVD9_IF2	LVD9_IF2 flag. This bit is the low-voltage status flag associated with the voltage level detect for the second high voltage 2.8 V I/O Flexray supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD9IE_IF2 bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_IF2] is also asserted, a system reset is generated, which clears VD9IE_IF2 and negate the interrupt request. If REE_VD9[LVD9_IF2] is asserted and a destructive reset is selected (via RES_VD9[LVD9_IF2] being 0), then a destructive reset is generated. If RES_VD9[LVD9_IF2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the second high voltage I/O Flexray supply.
21 LVD9_IF	LVD9_IF flag. This bit is the low-voltage status flag associated with the voltage level detect for the first high voltage 2.8 V I/O Flexray supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. If the VD9IE_IF bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_IF] is also asserted, a system reset is generated, which clears VD9IE_IF and negates the interrupt request. If REE_VD9[LVD9_IF] is asserted and a destructive reset is selected (via RES_VD9[LVD9_IF] being 0), then a destructive reset is generated. If RES_VD9[LVD9_IF] is set, then a functional reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the first high voltage I/O Flexray supply.
22 LVD9_IJ	LVD9_IJ flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O JTAG supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. If the VD9IE_IJ bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_IJ] is also asserted, a system reset is generated, which clears VD9IE_IJ and negates the interrupt request. If REE_VD9[LVD9_IJ] is asserted, a POR reset is generated.

Table continues on the next page...

PMCDIG_EPR_VD9 field descriptions (continued)

Field	Description
	This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O JTAG supply.
23 LVD9_IM	LVD9_IM flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V I/O main supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. If the VD9IE_IM bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_IM] is also asserted, a system reset is generated, which clears VD9IE_IM and negates the interrupt request. If REE_VD9[LVD9_A] is asserted, a POR reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O main supply.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 LVD9_F	LVD9_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V flash supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. If the VD9IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_F] is also asserted, a system reset is generated, which clears VD9IE_F and negates the interrupt request. If REE_VD9[LVD9_F] is asserted, a POR reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage flash supply.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LVD9_EBI	LVD9_EBI flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V External Bus Interface (EBI) supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD9IE_EBI bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_EBI] is also asserted, a system reset is generated, which clears VD9IE_EBI and negates the interrupt request. If REE_VD9[LVD9_EBI] is asserted, a POR reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage External Bus Interface supply.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD9_C	LVD9_C flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 2.8 V cold point supply. It is asserted asynchronously when the supply falls below its corresponding LVD

Table continues on the next page...

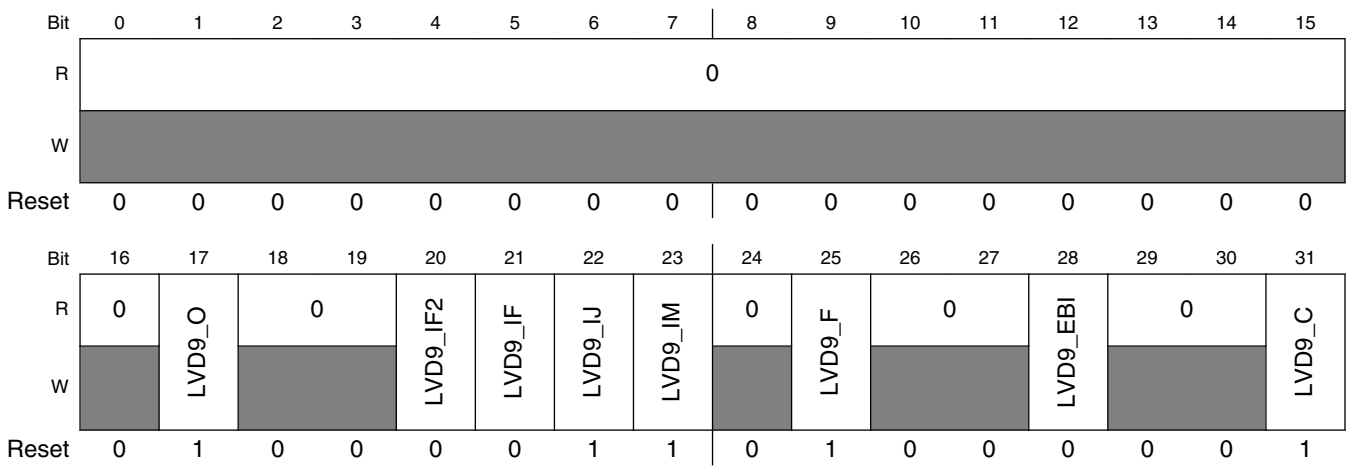
PMCDIG_EPR_VD9 field descriptions (continued)

Field	Description
	threshold, and clears when the supply rises above its corresponding LVD threshold and a one is written to this bit location. If the VD9IE_C bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD9[LVD9_C] is also asserted, a system reset is generated, which clears VD9IE_C and negates the interrupt request. If REE_VD9[LVD9_C] is asserted, a POR reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage cold point supply.

63.2.21 Reset Event Enable Register (PMCDIG_REE_VD9)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event.

Address: 0h base + 94h offset = 94h



PMCDIG_REE_VD9 field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 LVD9_O	LVD9_O reset enable. This bit defines whether an LVD assertion on the supply of the OSC generates a POR reset. 0 Disabled. LVD assertion on the supply of the OSC does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the OSC causes a POR reset.

Table continues on the next page...

PMCDIG_REE_VD9 field descriptions (continued)

Field	Description
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 LVD9_IF2	LVD9_IF2 reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the second set of Flexray pins generates a system reset. The RES_VD9[LVD9_IF2] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the second set of Flexray pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the second set of Flexray pins causes system reset.
21 LVD9_IF	LVD9_IF reset enable. This bit defines whether an LVD assertion on the supply of the first I/O segment that contains the Flexray pins generates a system reset. The RES_VD9[LVD9_IF] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the first Flexray pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the first Flexray pins causes system reset.
22 LVD9_IJ	LVD9_IJ reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the JTAG pins generates a POR reset. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins causes a POR reset.
23 LVD9_IM	LVD9_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a POR reset. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes a POR reset.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 LVD9_F	LVD9_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a POR reset. 0 Disabled. LVD assertion on the supply of the flash does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the flash causes a POR reset.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LVD9_EBI	LVD9_EBI reset enable.

Table continues on the next page...

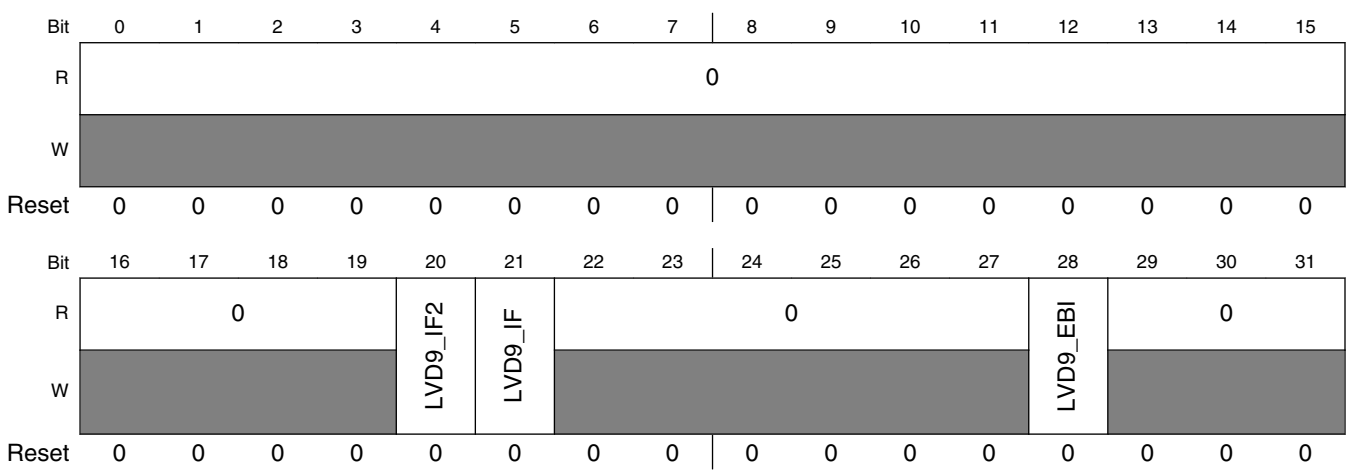
PMCDIG_REE_VD9 field descriptions (continued)

Field	Description
	This bit defines whether an LVD assertion on the supply of the External Bus Interface (EBI) generates a POR reset. 0 Disabled. LVD assertion on the supply of the External Bus Interface does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the External Bus Interface causes a POR reset.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LVD9_C	LVD9_C reset enable. This bit defines whether an LVD assertion on the supply of the cold point generates a POR reset. 0 Disabled. LVD assertion on the supply of the cold point does not cause a POR reset. 1 Enabled. LVD assertion on the supply of the cold point causes a POR reset.

63.2.22 Reset Event Select Register (PMCDIG_RES_VD9)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. All of the LVD9 signals are used to generate a POR except for LVD9_IF, LVD9_IF2, and LVD9_EBI. The LVD9_IF, LVD9_IF2, and LVD9_EBI are the only events at this voltage level that can cause a functional or destructive reset. All the others cause a POR (if enabled via the corresponding REE bit) or do not cause a reset.

Address: 0h base + 98h offset = 98h



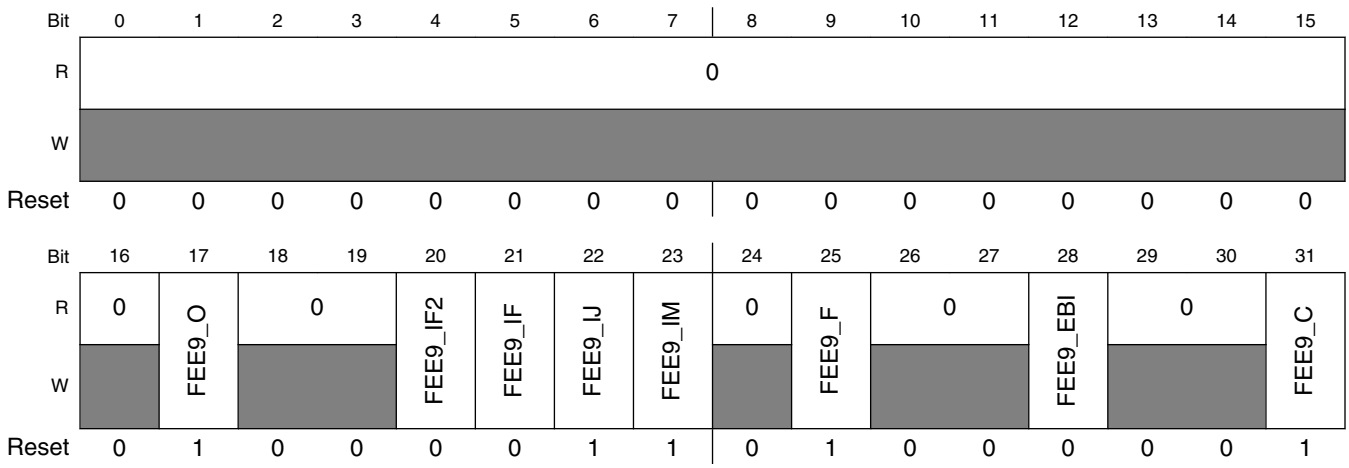
PMCDIG_RES_VD9 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 LVD9_IF2	LVD9_IF2 reset event select. This bit defines whether an LVD assertion on the supply of the second I/O segment that contains the Flexray pins generates a destructive reset or a functional reset. The REE_VD9[LVD9_IF2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the second I/O segment that contains the Flexray pins causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the second I/O segment that contains the Flexray pins causes a functional system reset.
21 LVD9_IF	LVD9_IF reset event select. This bit defines whether an LVD assertion on the supply of the first I/O segment that contains the Flexray pins generates a destructive reset or a functional reset. The REE_VD9[LVD9_IF] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the first I/O segment that contains the Flexray pins causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the first I/O segment that contains the Flexray pins causes a functional system reset.
22–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LVD9_EBI	LVD9_EBI reset event select. This bit defines whether an LVD assertion on the supply of the first I/O segment that contains the External Bus Interface pins generates a destructive reset or a functional reset. The REE_VD9[LVD9_EBI] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the first I/O segment that contains the External Bus Interface pins causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the first I/O segment that contains the External Bus Interface pins causes a functional system reset.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.23 FCCU Event Enable VD9 (PMCDIG_FEE_VD9)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + 9Ch offset = 9Ch



PMCDIG_FEE_VD9 field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 FEE9_O	FEE9_O FCCU event enable. This bit defines whether an LVD assertion on the supply of the OSC generates an FCCU event. 0 Disabled. LVD assertion on the supply of the OSC does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the OSC causes an FCCU event.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 FEE9_IF2	FEE9_IF2 FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the second set of Flexray pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the second set of Flexray pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the second set of Flexray pins causes an FCCU event.
21 FEE9_IF	FEE9_IF FCCU event enable.

Table continues on the next page...

PMCDIG_FEE_VD9 field descriptions (continued)

Field	Description
	This bit defines whether an LVD assertion on the supply of the first I/O segment that contains the Flexray pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the first Flexray pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the first Flexray pins causes an FCCU event.
22 FEE9_IJ	FEE9_IJ FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the JTAG pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the JTAG pins causes an FCCU event.
23 FEE9_IM	FEE9_IM FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes an FCCU event.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FEE9_F	FEE9_F FCCU event enable. This bit defines whether an LVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. LVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the flash causes an FCCU event.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 FEE9_EBI	FEE9_EBI FCCU event enable. This bit defines whether an LVD assertion on the supply of the External Bus Interface (EBI) generates an FCCU event. 0 Disabled. LVD assertion on the supply of the External Bus Interface does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the External Bus Interface causes an FCCU event.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE9_C	FEE9_C FCCU event enable. This bit defines whether an LVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. LVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the cold point causes an FCCU event.

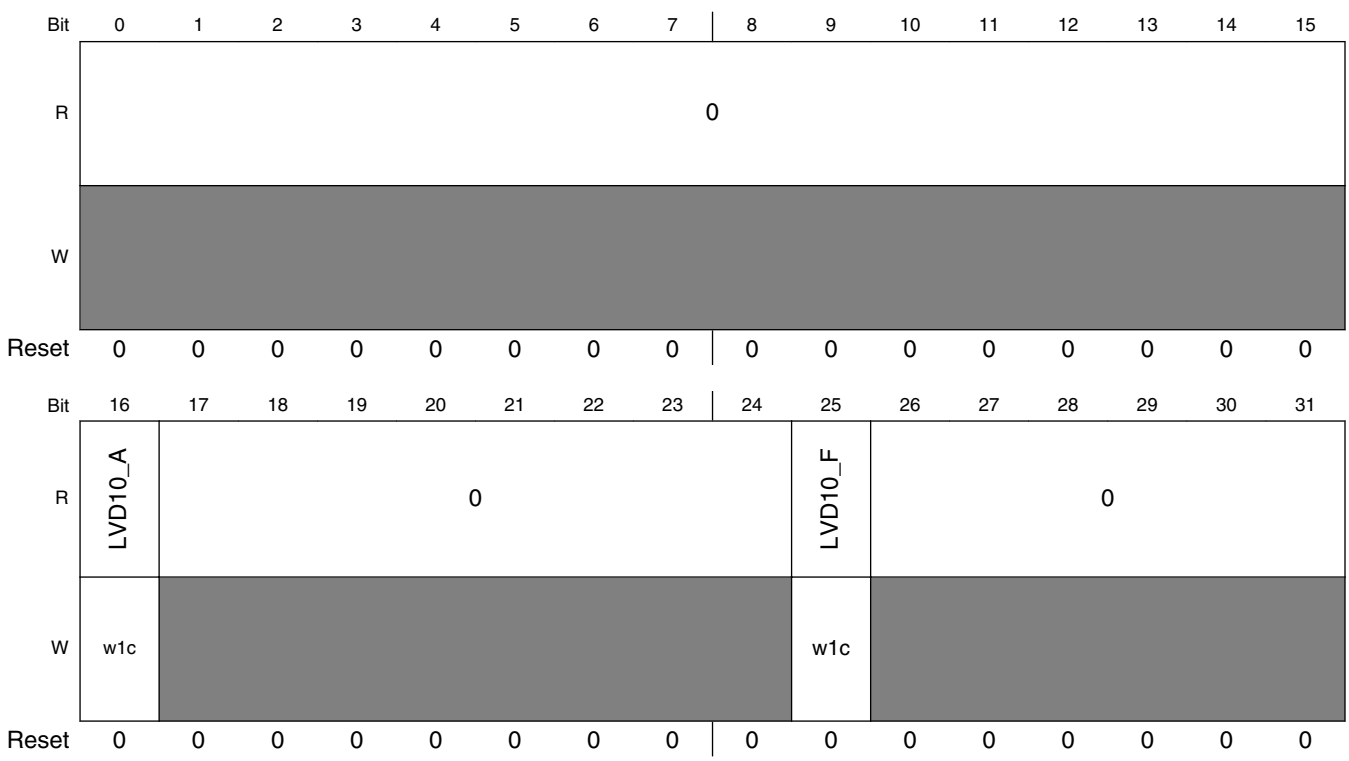
63.2.24 LVD295 Event Pending Register (PMCDIG_EPR_VD10)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

NOTE

These bits are cleared out of initial power up POR, but are not reset during other LVD, HVD, and POR events. They must be cleared by software during initialization by writing 1 to the EPR_VD9 field. The field values are retained until the core voltage is less than 0.8V).

Address: 0h base + A0h offset = A0h



PMCDIG_EPR_VD10 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LVD10_A	LVD10_A flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V ADC supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and

Table continues on the next page...

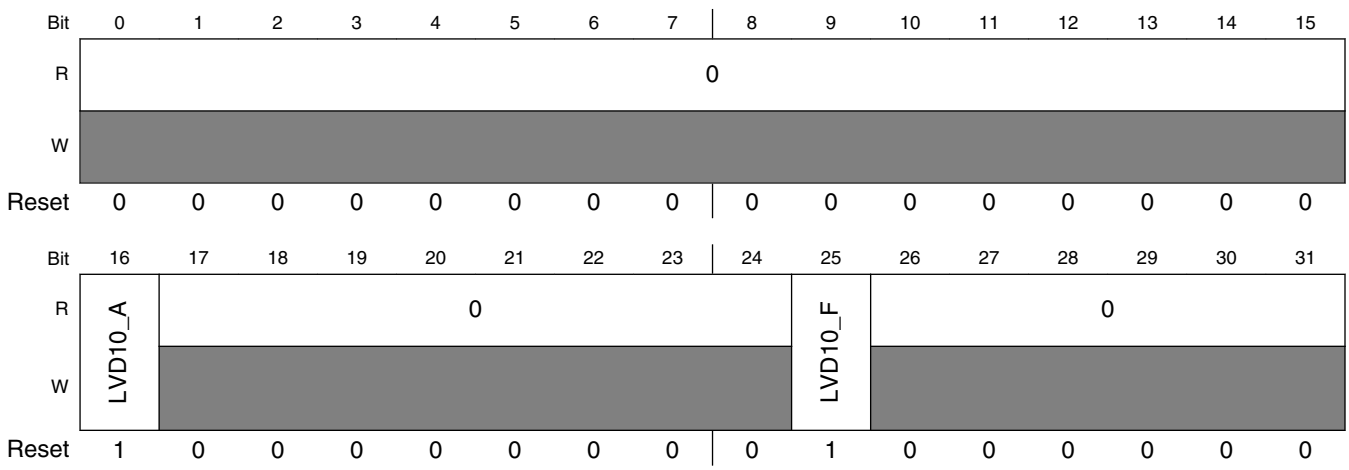
PMCDIG_EPR_VD10 field descriptions (continued)

Field	Description
	clears when a one is written to this bit location. If the VD10IE_A bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD10[LVD10_A] is also asserted, a POR reset is generated, which clears VD10IE_A and negates the interrupt request. If REE_VD10[LVD10_A] is asserted, a POR reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage ADC supply.
17–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 LVD10_F	LVD10_F flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3 V flash supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD10IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD10[LVD10_F] is also asserted, a POR reset is generated, which clears VD10IE_F and negates the interrupt request. If REE_VD10[LVD10_F] is asserted, a POR reset is generated. This field is indeterminate after reset, so it must be cleared during initialization by writing 1 to this location. This field may be cleared if the LV supply is below 0.9 V. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage flash supply.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.25 Reset Event Enable Register (PMCDIG_REE_VD10)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + A4h offset = A4h



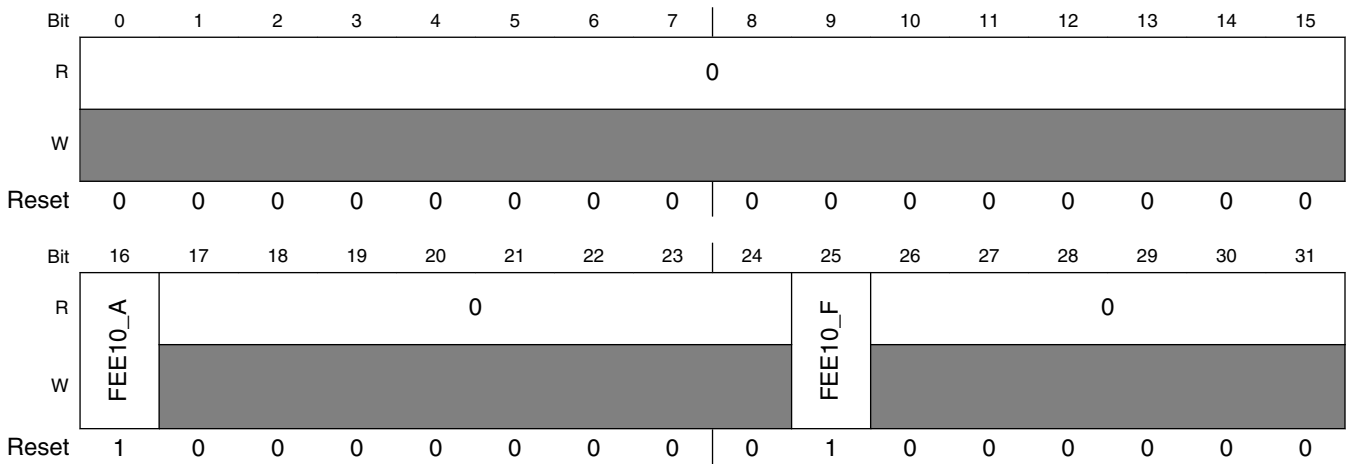
PMCDIG_REE_VD10 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LVD10_A	LVD10_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
17–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 LVD10_F	LVD10_F reset enable. This bit defines whether an LVD assertion on the supply of the flash generates a system reset. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset. 1 Enabled. LVD assertion on the supply of the flash causes system reset.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.26 FCCU Event Enable Register (PMCDIG_FEE_VD10)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + ACh offset = ACh



PMCDIG_FEE_VD10 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FEE10_A	FEE10_A FCCU event enable. This bit defines whether an LVD assertion on the supply of the ADC generates an FCCU event. 0 Disabled. LVD assertion on the supply of the ADC does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the ADC causes an FCCU event.
17–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FEE10_F	FEE10_F FCCU event enable. This bit defines whether an LVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. LVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the flash causes an FCCU event.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.27 HVD360 Event Pending Register (PMCDIG_EPR_VD12)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									HVD12_F	0					
W										w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_EPR_VD12 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 HVD12_F	HVD12_F flag. This bit is the high-voltage status flag associated with the voltage level detect for the high voltage 3.6 V flash supply. It is asserted asynchronously when the supply rises above its corresponding HVD threshold, and clears when a one is written to this bit location. If the VD12IE_F bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD12[HVD12_F] is also asserted, a system reset is generated, which clears VD12IE_F and negates the interrupt request. If REE_VD12[HVD12_F] is asserted and a destructive reset is selected (via RES_VD12[HVD12_F] being 0), then a destructive reset is generated. If RES_VD12[HVD12_F] is set, then a functional reset is generated.

Table continues on the next page...

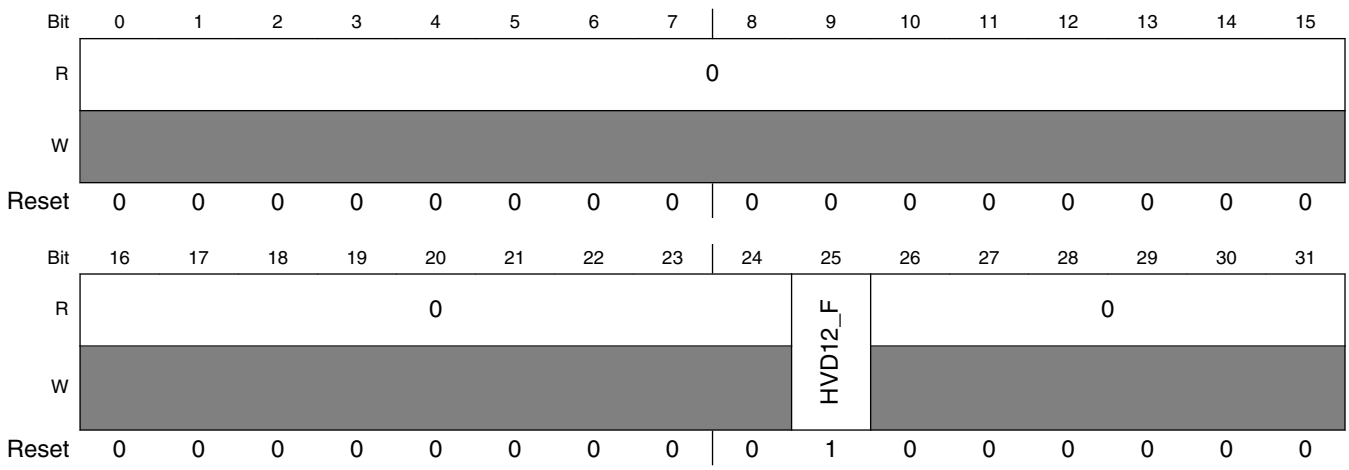
PMCDIG_EPR_VD12 field descriptions (continued)

Field	Description
	0 Currently no occurrence. 1 HVD occurrence detected on the high voltage flash supply.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.28 Reset Event Enable Register (PMCDIG_REE_VD12)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + C4h offset = C4h



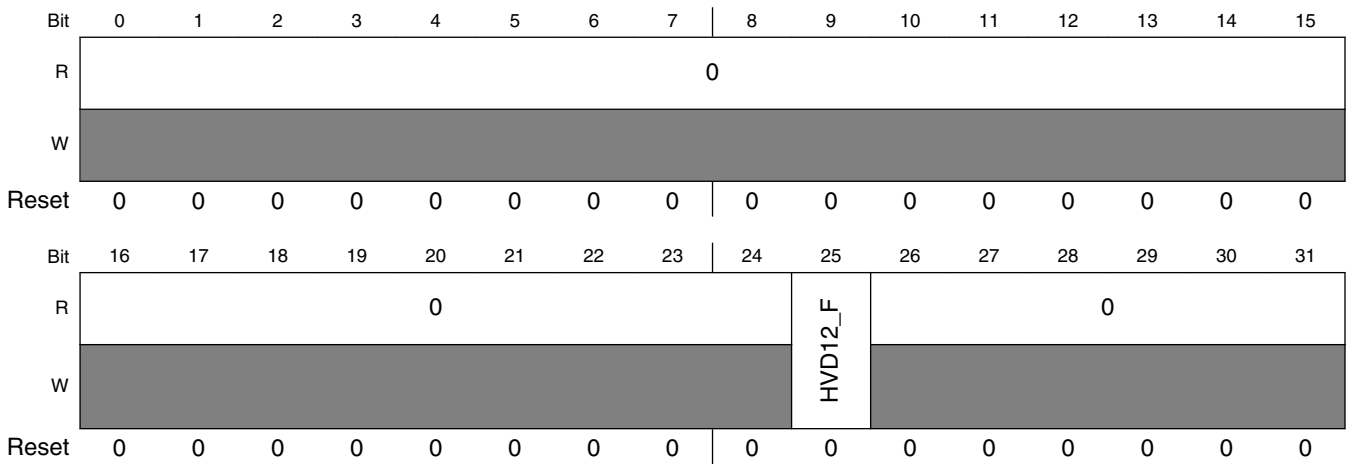
PMCDIG_REE_VD12 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 HVD12_F	HVD12_F reset enable. This bit defines whether an HVD assertion on the supply of the flash generates a system reset. The RES_VD12[HVD12_F] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the flash does not cause system reset. 1 Enabled. HVD assertion on the supply of the flash causes system reset.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.29 Reset Event Select Register (PMCDIG_RES_VD12)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + C8h offset = C8h



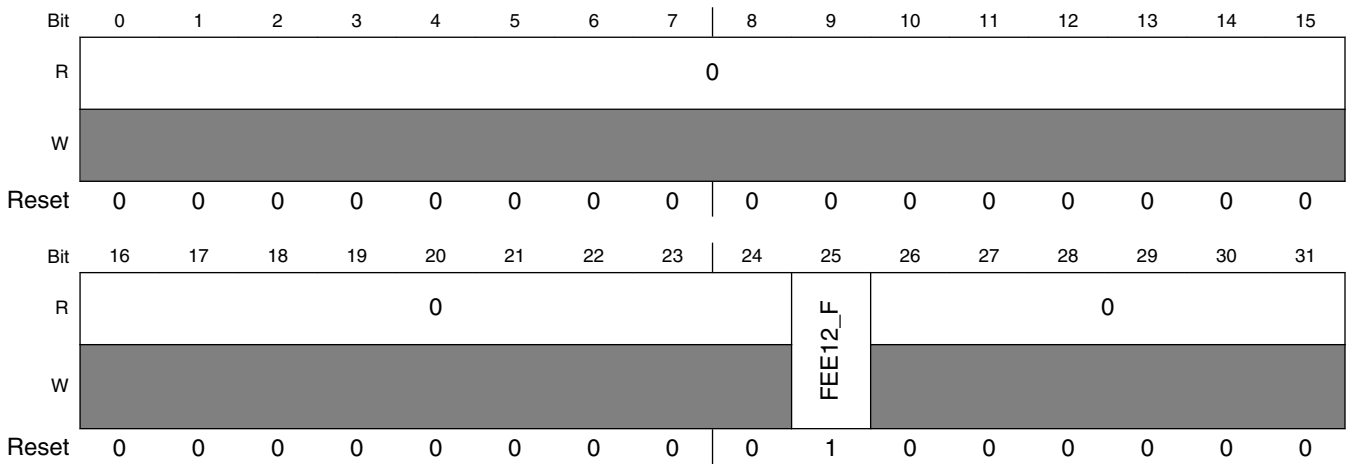
PMCDIG_RES_VD12 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 HVD12_F	HVD12_F reset event select. This bit defines whether an HVD assertion on the supply of the flash generates a destructive reset or a functional reset. The REE_VD12[HVD12_F] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the flash causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the flash causes a functional system reset.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.30 FCCU Event Enable Register (PMCDIG_FEE_VD12)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + CCh offset = CCh



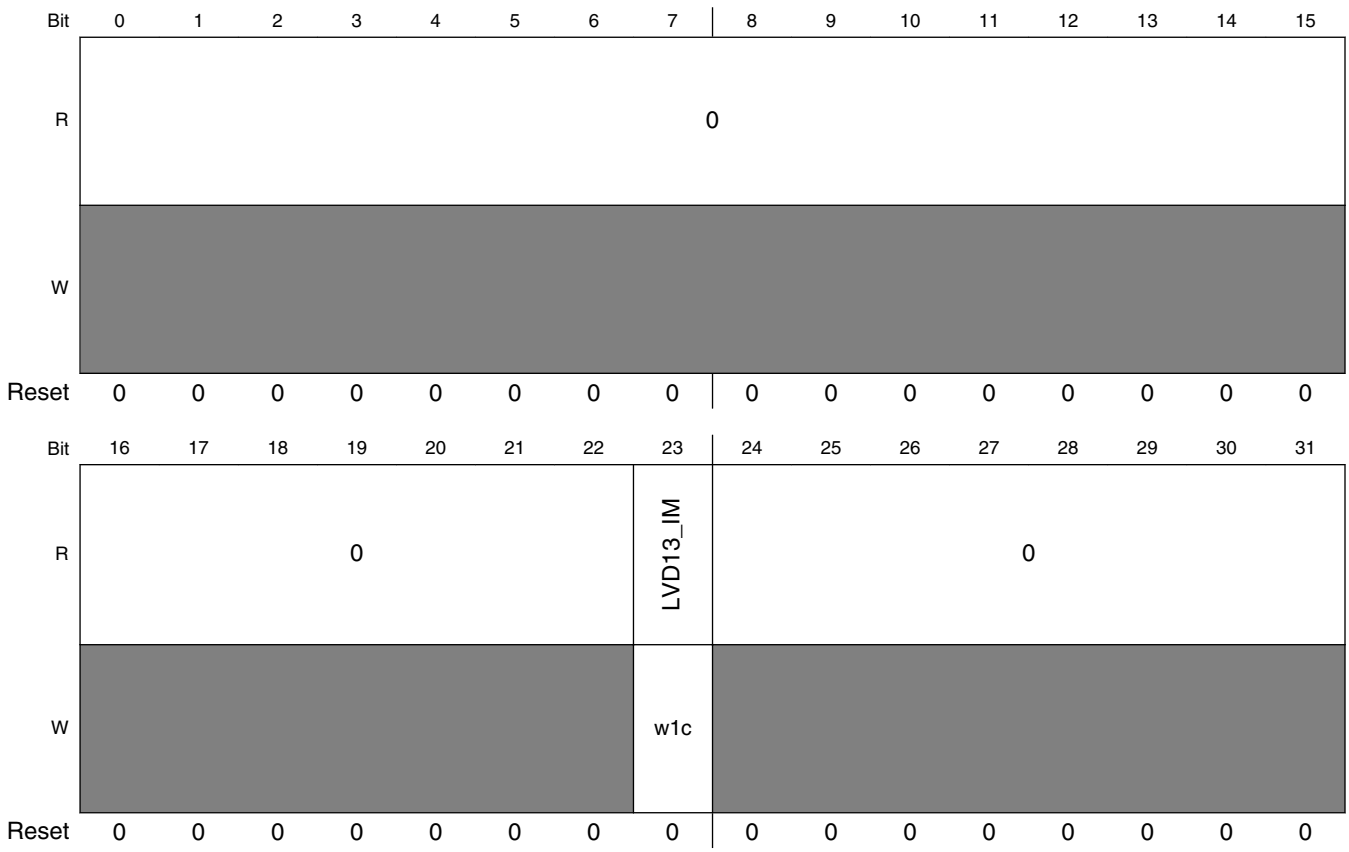
PMCDIG_FEE_VD12 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FEE12_F	FEE12_F FCCU event enable. This bit defines whether an HVD assertion on the supply of the flash generates an FCCU event. 0 Disabled. HVD assertion on the supply of the flash does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the flash causes an FCCU event.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.31 Event Pending Register (PMCDIG_EPR_VD13)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + D0h offset = D0h



PMCDIG_EPR_VD13 field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 LVD13_IM	LVD13_IM flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 3.6 V I/O main supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD13IE_IM bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD13[LVD13_IM] is also asserted, a system reset is generated, which clears VD13IE_IM and negates the interrupt request. If REE_VD13[LVD13_IM] is asserted and a

Table continues on the next page...

PMCDIG_EPR_VD13 field descriptions (continued)

Field	Description
	destructive reset is selected (via RES_VD13[LVD13_IM] being 0), then a destructive reset is generated. If RES_VD13[LVD13_IM] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O main supply.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.32 Reset Event Enable Register (PMCDIG_REE_VD13)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							LVD13_IM	0							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

PMCDIG_REE_VD13 field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 LVD13_IM	LVD13_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a system reset. The RES_VD13[LVD13_IM] bit determines whether the reset is functional or destructive.

Table continues on the next page...

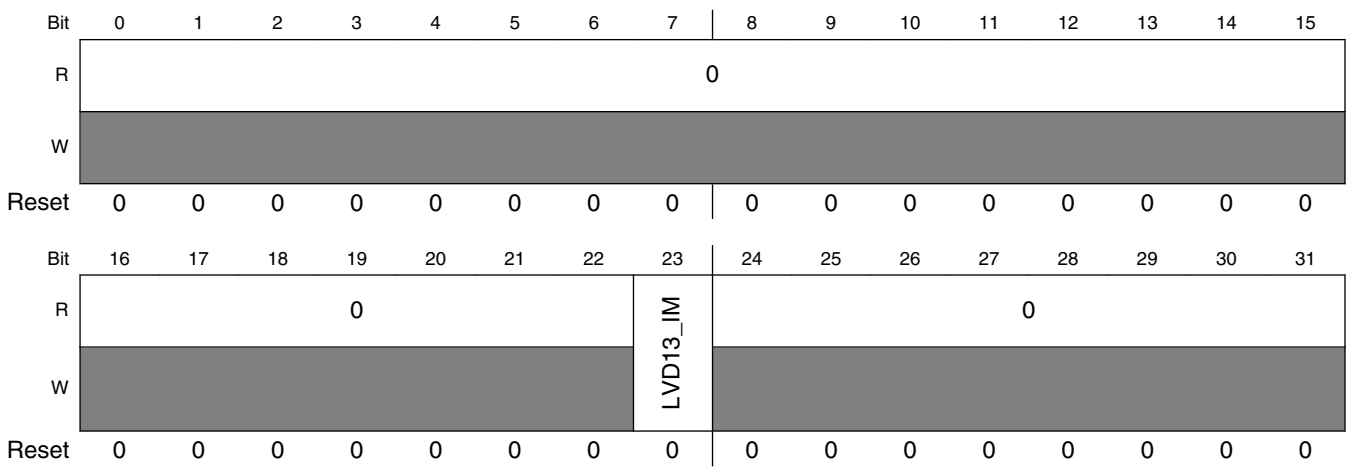
PMCDIG_REE_VD13 field descriptions (continued)

Field	Description
	0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes system reset.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.33 Reset Event Select Register (PMCDIG_RES_VD13)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + D8h offset = D8h



PMCDIG_RES_VD13 field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 LVD13_IM	LVD13_IM reset event select. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates a destructive reset or a functional reset. The REE_VD13[LVD13_IM] bit determines whether the reset event is enabled.

Table continues on the next page...

PMCDIG_RES_VD13 field descriptions (continued)

Field	Description
	0 Destructive Reset Generated. LVD assertion on the supply of the I/O segment that contains the main pins causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the I.O segment that contains the main pins causes a functional system reset.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.34 FCCU Event Enable Register (PMCDIG_FEE_VD13)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							FEE13_IM	0							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

PMCDIG_FEE_VD13 field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEE13_IM	FEE13_IM FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O segment that contains the main pins generates an FCCU event. 0 Disabled. LVD assertion on the supply of the I/O segment that contains the main pins does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O segment that contains the main pins causes an FCCU event.

Table continues on the next page...

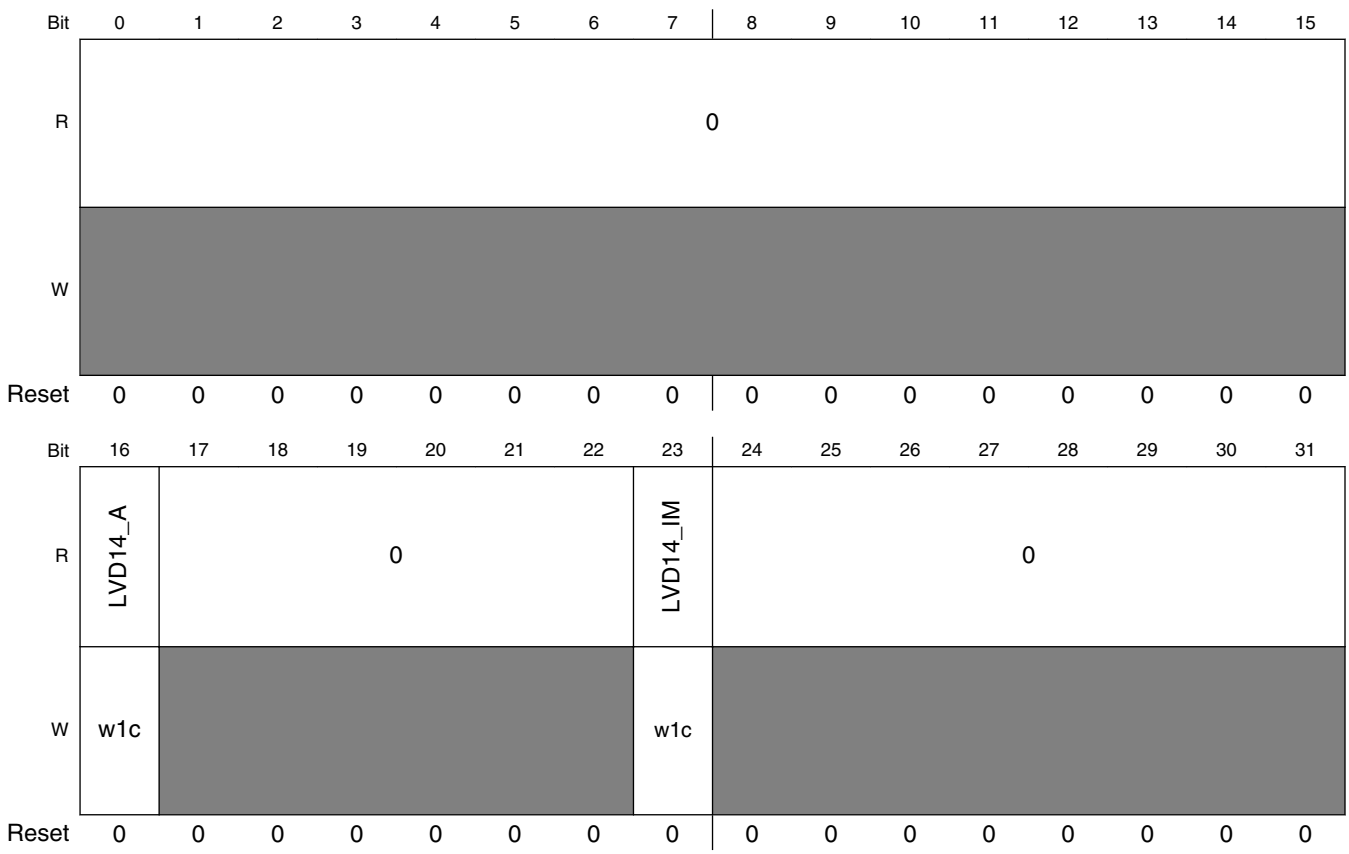
PMCDIG_FEE_VD13 field descriptions (continued)

Field	Description
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.35 Event Pending Register (PMCDIG_EPR_VD14)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + E0h offset = E0h



PMCDIG_EPR_VD14 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

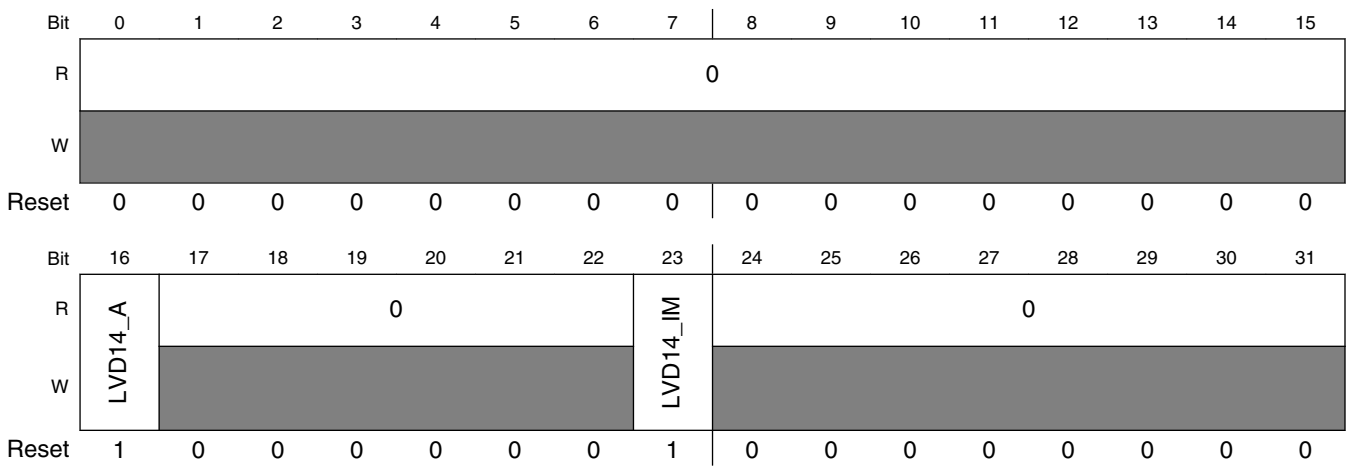
PMCDIG_EPR_VD14 field descriptions (continued)

Field	Description
16 LVD14_A	LVD14_A flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V ADC supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD14IE_A bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD14[LVD14_A] is also asserted, a system reset is generated, which clears VD14IE_A and negates the interrupt request. If REE_VD14[LVD14_A] is asserted and a destructive reset is selected (via RES_VD14[LVD14_A] being 0), then a destructive reset is generated. If RES_VD14[LVD14_A] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage ADC supply.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 LVD14_IM	LVD14_IM flag. This bit is the low-voltage status flag associated with the voltage level detect for the high voltage 4 V I/O Main supply. It is asserted asynchronously when the supply falls below its corresponding LVD threshold, and clears when a one is written to this bit location. If the VD14IE_IM bit is also asserted, a low-voltage interrupt is sent to the CPU. If REE_VD14[LVD14_IM] is also asserted, a system reset is generated, which clears VD14IE_IM and negates the interrupt request. If REE_VD14[LVD14_IM] is asserted and a destructive reset is selected (via RES_VD14[LVD14_IM] being 0), then a destructive reset is generated. If RES_VD14[LVD14_IM] is set, then a functional reset is generated. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage I/O Main supply.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.36 Reset Event Enable Register (PMCDIG_REE_VD14)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + E4h offset = E4h



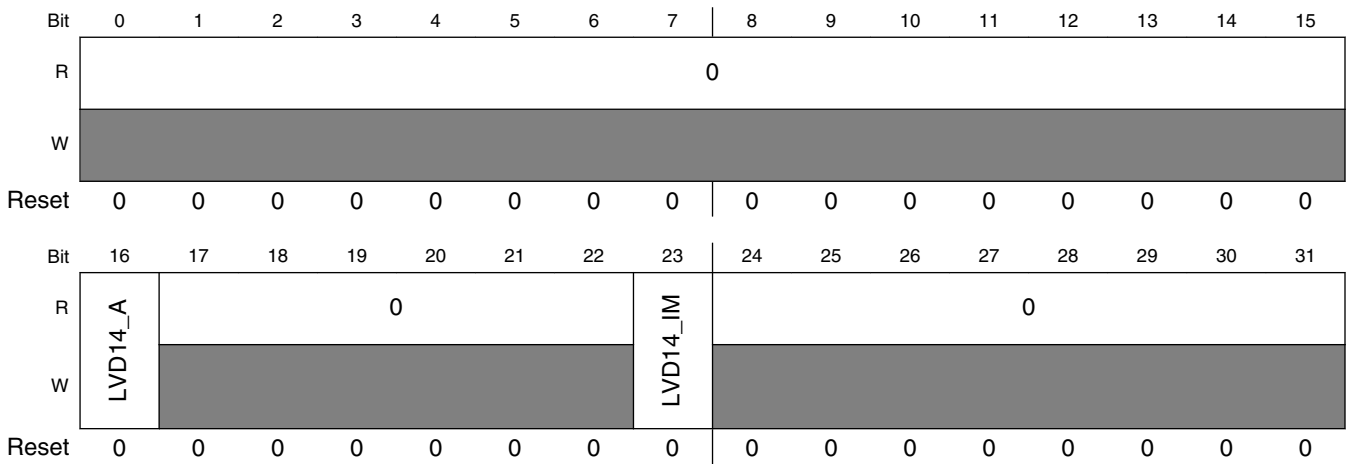
PMCDIG_REE_VD14 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LVD14_A	LVD14_A reset enable. This bit defines whether an LVD assertion on the supply of the ADC generates a system reset. The RES_VD14[LVD14_A] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the supply of the ADC causes system reset.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 LVD14_IM	LVD14_IM reset enable. This bit defines whether an LVD assertion on the supply of the I/O Main generates a system reset. The RES_VD14[LVD14_IM] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the I/O Main does not cause system reset. 1 Enabled. LVD assertion on the supply of the I/O Main causes system reset.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.37 Reset Event Select Register (PMCDIG_RES_VD14)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + E8h offset = E8h



PMCDIG_RES_VD14 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LVD14_A	LVD14_A reset event select. This bit defines whether an LVD assertion on the supply of the ADC generates a destructive reset or a functional reset. The REE_VD14[LVD14_A] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the ADC causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the ADC causes a functional system reset.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 LVD14_IM	LVD14_IM reset event select. This bit defines whether an LVD assertion on the supply of the IO Main generates a destructive reset or a functional reset. The REE_VD14[LVD14_IM] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the I/O Main causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the I/O Main causes a functional system reset.

Table continues on the next page...

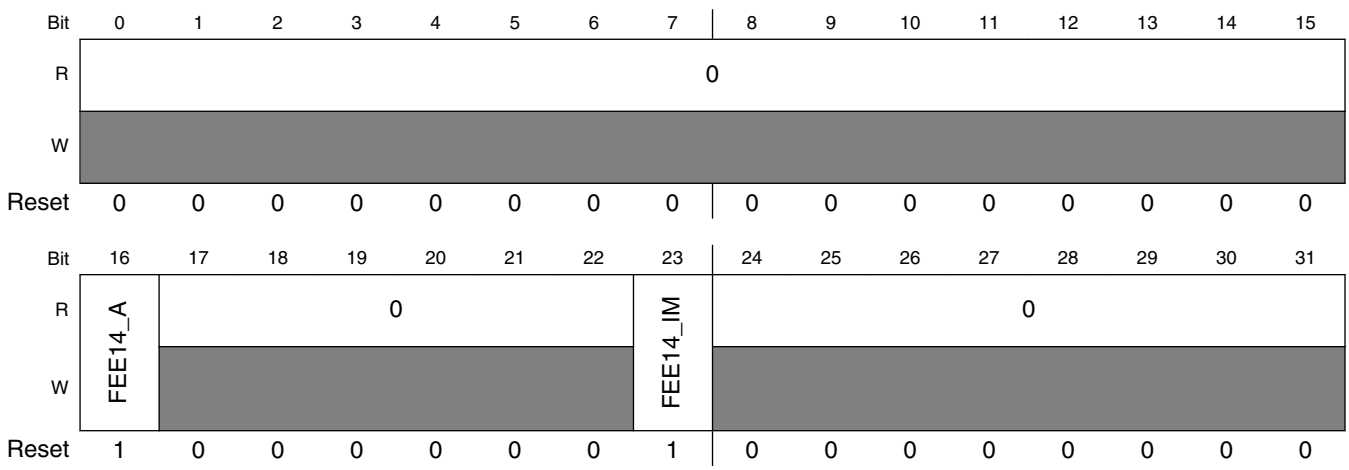
PMCDIG_RES_VD14 field descriptions (continued)

Field	Description
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.38 FCCU Event Enable Register (PMCDIG_FEE_VD14)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + ECh offset = ECh



PMCDIG_FEE_VD14 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FEE14_A	FEE14_A FCCU event enable. This bit defines whether an LVD assertion on the supply of the ADC generates an FCCU event. 0 Disabled. LVD assertion on the supply of the ADC does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the ADC causes an FCCU event.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEE14_IM	FEE14_IM FCCU event enable. This bit defines whether an LVD assertion on the supply of the I/O Main generates an FCCU event.

Table continues on the next page...

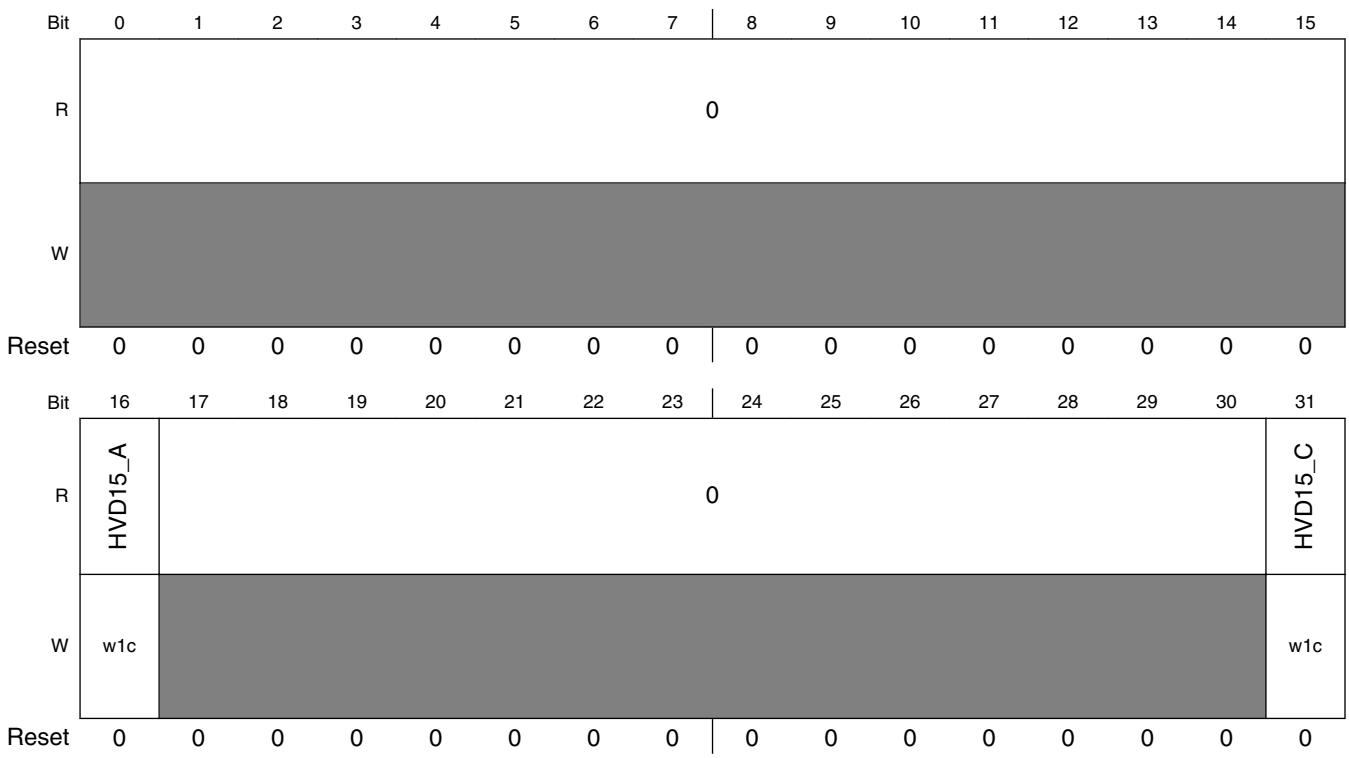
PMCDIG_FEE_VD14 field descriptions (continued)

Field	Description
	0 Disabled. LVD assertion on the supply of the I/O Main does not cause an FCCU event. 1 Enabled. LVD assertion on the supply of the I/O Main causes an FCCU event.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.39 Event Pending Register (PMCDIG_EPR_VD15)

This Event Pending Register indicates the present and past state of the voltage detect signals. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set asynchronously. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + F0h offset = F0h



PMCDIG_EPR_VD15 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

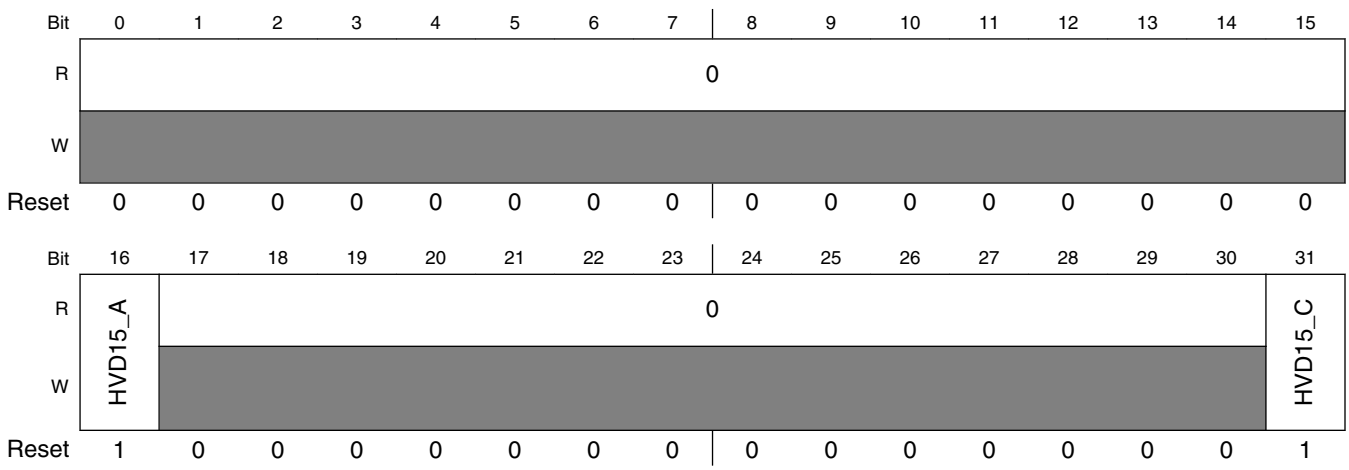
PMCDIG_EPR_VD15 field descriptions (continued)

Field	Description
16 HVD15_A	HVD15_A flag. This bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V ADC supply. It is asserted asynchronously when the supply rises above its corresponding HVD threshold, and clears when a one is written to this bit location. If the VD15IE_A bit is also asserted, a high-voltage interrupt is sent to the CPU. If REE_VD15[HVD15_A] is also asserted, a system reset is generated, which clears VD15IE_A and negate the interrupt request. If REE_VD15[HVD15_A] is asserted and a destructive reset is selected (via RES_VD15[HVD15_A] being 0), then a destructive reset is generated. If RES_VD15[HVD15_A] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage ADC supply.
17–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD15_C	HVD15_C flag. This bit is the high-voltage status flag associated with the voltage level detect for the high voltage 6 V cold point supply. It is asserted asynchronously when the supply rises above its corresponding HVD threshold, and clears when a one is written to this bit location. If the VD15IE_C bit is also asserted, a high-voltage interrupt is sent to the CPU. If REE_VD15[HVD15_C] is also asserted, a system reset is generated, which clears VD15IE_C and negates the interrupt request. If REE_VD15[HVD15_C] is asserted and a destructive reset is selected (via RES_VD15[HVD15_C] being 0), then a destructive reset is generated. If RES_VD15[HVD15_C] is set, then a functional reset is generated. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage cold point supply.

63.2.40 Reset Event Enable Register (PMCDIG_REE_VD15)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared) .

Address: 0h base + F4h offset = F4h



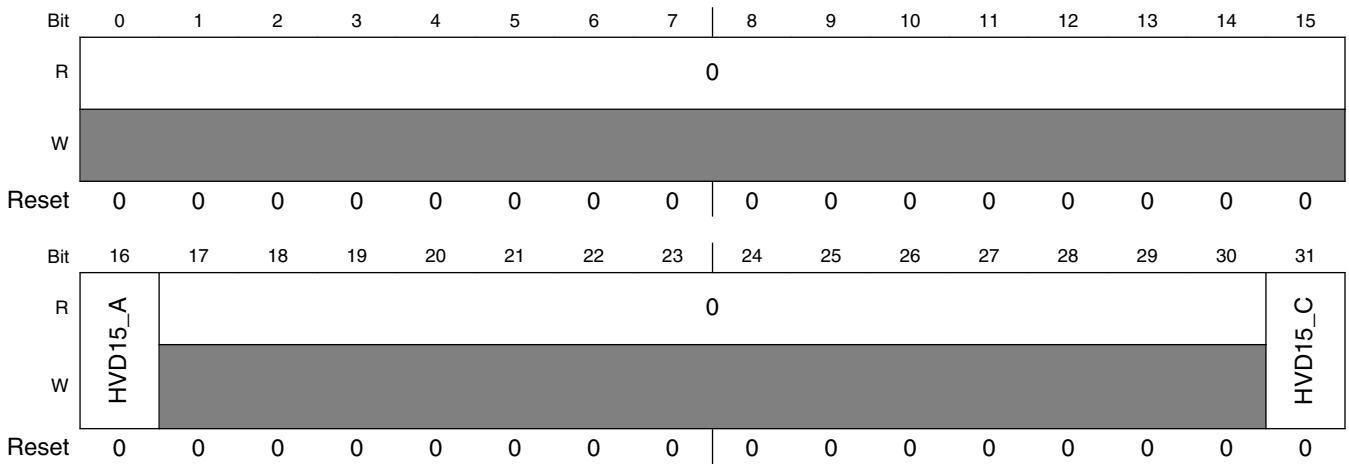
PMCDIG_REE_VD15 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 HVD15_A	HVD15_A reset enable. This bit defines whether an HVD assertion on the supply of the ADC generates a system reset. The RES_VD15[HVD15_A] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the ADC does not cause system reset. 1 Enabled. HVD assertion on the supply of the ADC causes system reset.
17–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD15_C	HVD15_C reset enable. This bit defines whether an HVD assertion on the supply of the cold point generates a system reset. The RES_VD15[HVD15_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

63.2.41 Reset Event Select Register (PMCDIG_RES_VD15)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Address: 0h base + F8h offset = F8h



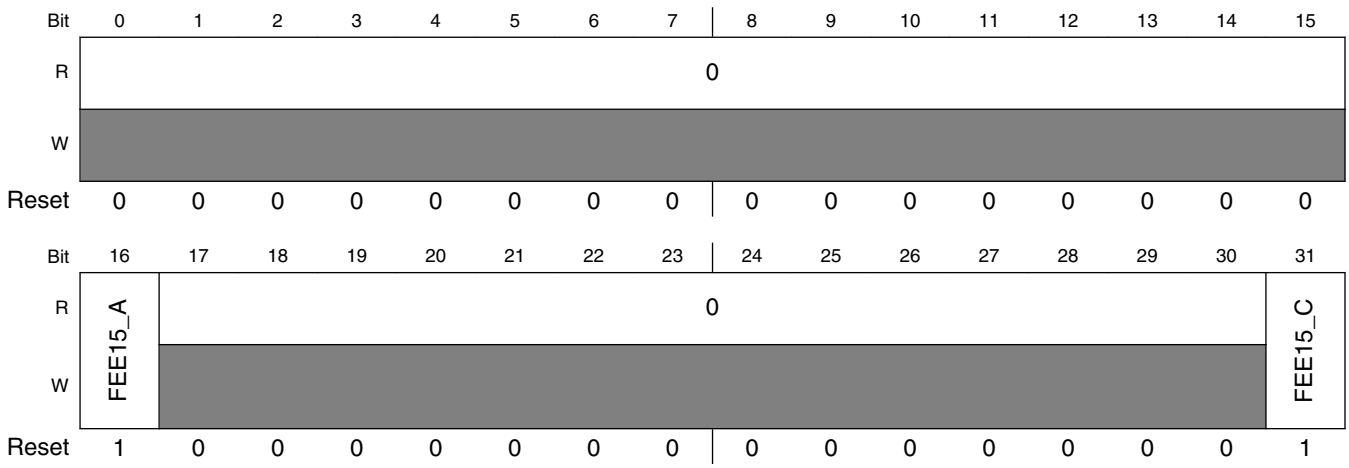
PMCDIG_RES_VD15 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 HVD15_A	HVD15_A reset event select. This bit defines whether an HVD assertion on the supply of the ADC generates a destructive reset or a functional reset. The REE_VD15[HVD15_A] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the ADC causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the ADC causes a functional system reset.
17–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVD15_C	HVD15_C reset event select. This bit defines whether an HVD assertion on the supply of the cold point generates a destructive reset or a functional reset. The REE_VD15[HVD15_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

63.2.42 FCCU Event Enable Register (PMCDIG_FEE_VD15)

This FCCU Event Enable Register controls whether the voltage detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Address: 0h base + FCh offset = FCh



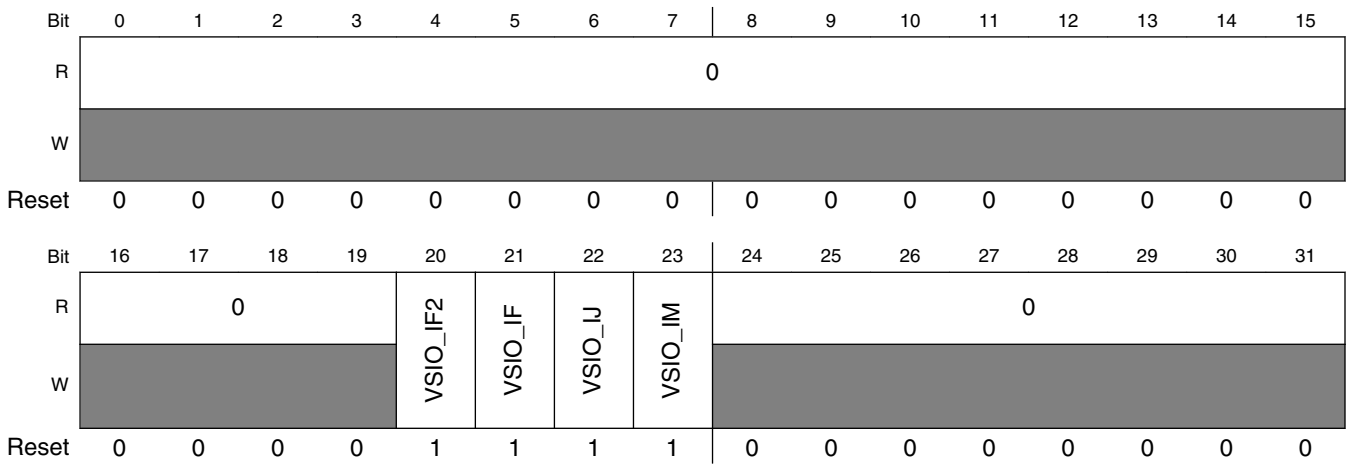
PMCDIG_FEE_VD15 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FEE15_A	FEE15_A FCCU event enable. This bit defines whether an HVD assertion on the supply of the ADC generates an FCCU event. 0 Disabled. HVD assertion on the supply of the ADC does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the ADC causes an FCCU event.
17–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE15_C	FEE15_C FCCU event enable. This bit defines whether an HVD assertion on the supply of the cold point generates an FCCU event. 0 Disabled. HVD assertion on the supply of the cold point does not cause an FCCU event. 1 Enabled. HVD assertion on the supply of the cold point causes an FCCU event.

63.2.43 Voltage Supply for I/O Segment Register (PMCDIG_VSIO)

The Voltage Supply for I/O Segments (VSIO) register indicates if the I/O segment indicated uses a 3.3V or 5.0V supply. These bits all reset to use a 5.0V supply.

Address: 0h base + 104h offset = 104h



PMCDIG_VSIO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 VSIO_IF2	VSIO_IF2 control. This bit is used to control the supply levels used by the second Flexray I/O segment. 0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
21 VSIO_IF	VSIO_IF control. This bit is used to control the supply levels used by the Flexray I/O segment. 0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
22 VSIO_IJ	VSIO_IJ control. This bit is used to control the supply levels used by the JTAG I/O segment. 0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
23 VSIO_IM	VSIO_IM control. This bit is used to control the supply levels used by the Main I/O segment.

Table continues on the next page...

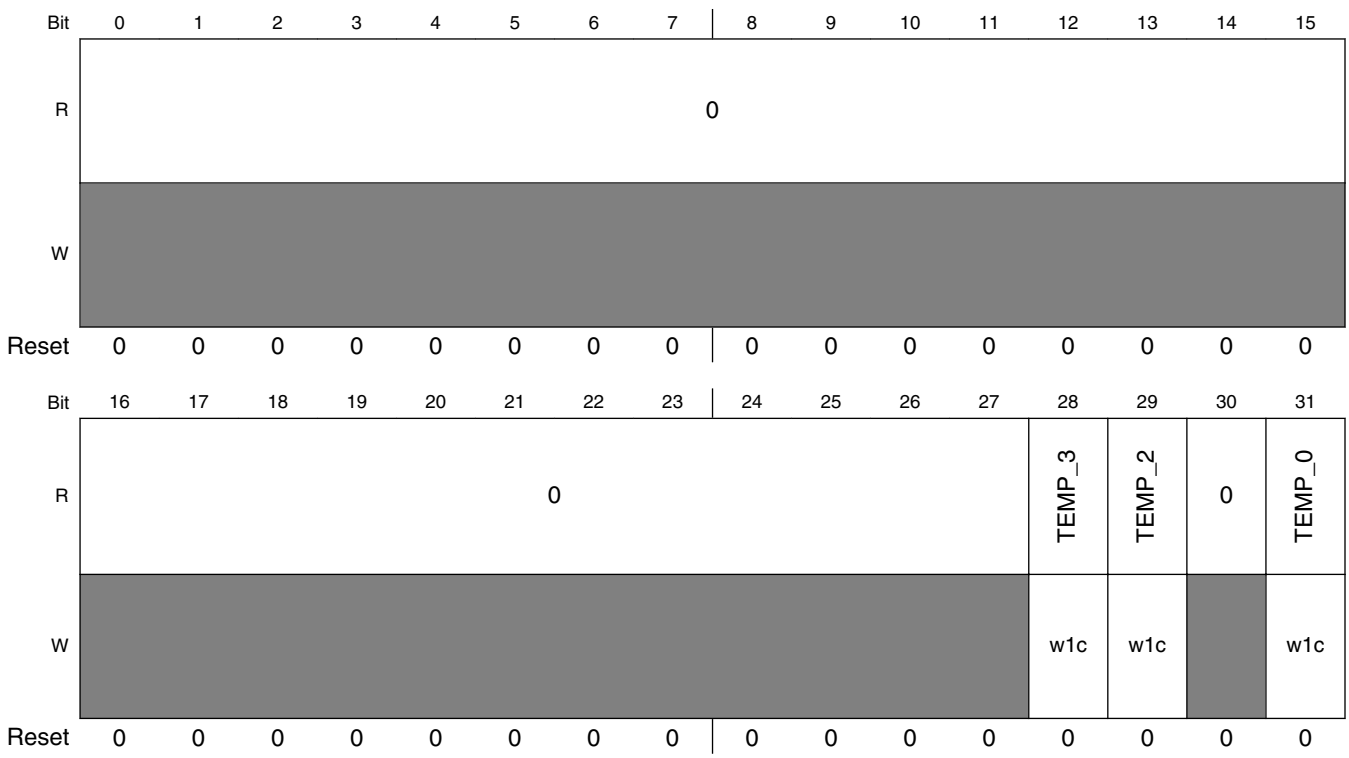
PMCDIG_VSIO field descriptions (continued)

Field	Description
	0 The I/O segment uses a 3.3V supply. 1 The I/O segment uses a 5.0V supply.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.2.44 Event Pending Register (PMCDIG_EPR_TD)

This Event Pending Register indicates the present and past state of the temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + 300h offset = 300h



PMCDIG_EPR_TD field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMCDIG_EPR_TD field descriptions (continued)

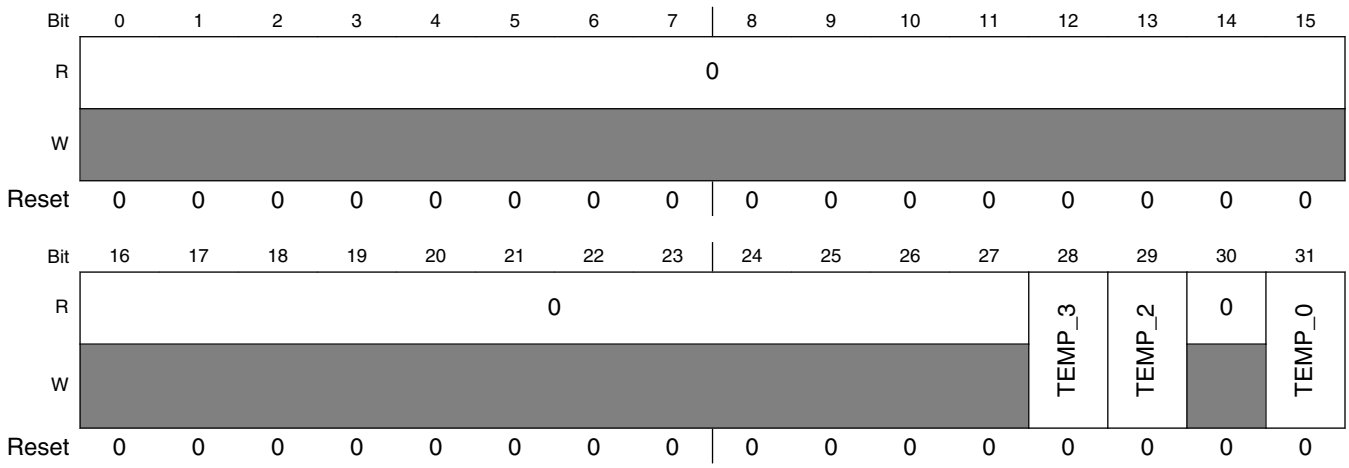
Field	Description
28 TEMP_3	TEMP_3 flag. This bit is the temperature status flag associated with the temperature for the high temperature sensor point (165C). It is asserted when the temperature exceeds its corresponding threshold, and clears when a one is written to this bit location. If the IETD_3 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_3] is also asserted, a system reset is generated, which clears IETD_3 and negates the interrupt request. If REE_TD[TEMP_3] is asserted and a destructive reset is selected (via RES_TD[TEMP_3] being 0), then a destructive reset is generated. If RES_TD[TEMP_3] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
29 TEMP_2	TEMP_2 flag. This bit is the temperature status flag associated with the temperature for the temperature sensor point (150C). It is asserted when the temperature exceeds its corresponding threshold, and clears when a one is written to this bit location. If the IETD_2 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_2 is also asserted, a system reset is generated, which clears IETD_2 and negates the interrupt request. If REE_TD[TEMP_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TEMP_0	TEMP_0 flag. This bit is the temperature status flag associated with the temperature for the cold temperature sensor point. It is asserted when the temperature exceeds its corresponding threshold, and clears when a one is written to this bit location. If the IETD_0 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_0] is also asserted, a system reset is generated, which clears IETD_0 and negates the interrupt request. If REE_TD[TEMP_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP_0] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.

63.2.45 Reset Event Enable Register (PMCDIG_REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Address: 0h base + 304h offset = 304h



PMCDIG_REE_TD field descriptions

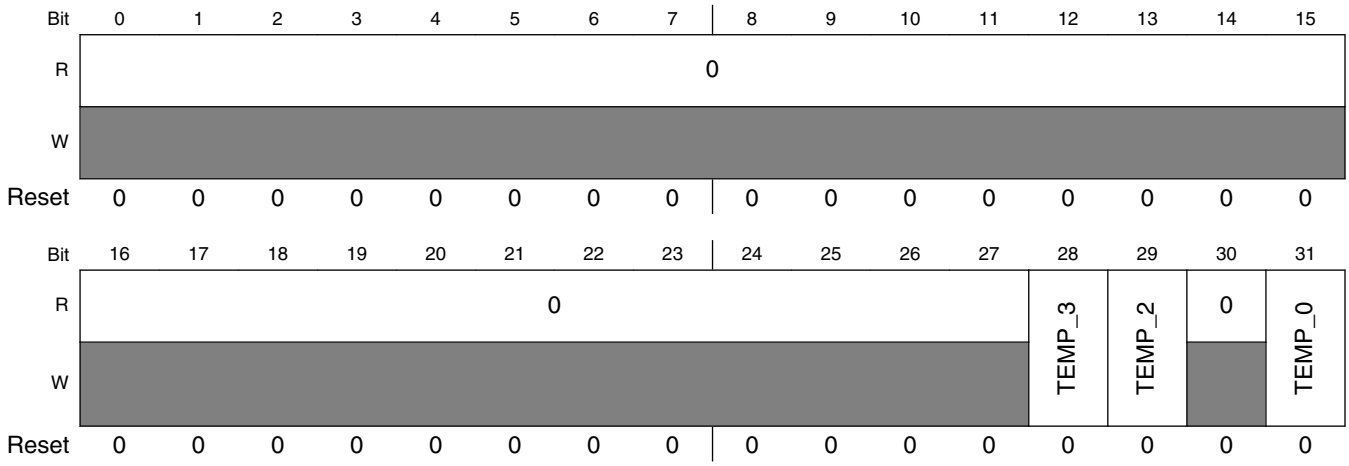
Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TEMP_3	TEMP_3 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
29 TEMP_2	TEMP_2 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_1] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TEMP_0	TEMP_0 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

63.2.46 Reset Event Select Register (PMCDIG_RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Address: 0h base + 308h offset = 308h



PMCDIG_RES_TD field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TEMP_3	TEMP_3 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_3] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
29 TEMP_2	TEMP_2 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TEMP_0	TEMP_0 reset event select. This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

63.2.47 Temperature Sensor Configuration Register (PMCDIG_CTL_TD)

The Temperature Sensor register (CTL_TD) contains two enables that must both be enabled in order for the Temperature Sensor to operate. It also includes two 4-bit signed trim adjustment register fields that are used to modify both TS trim values (one bus for overtemperature and one bus for undertemperature). Finally, this register includes Interrupt Enables (IE) for each of the Temperature Sensor EPR bits.

The use case for the TRIM_ADJ_OVER and TRIM_ADJ_UNDER fields would be to introduce additional hysteresis into the Temperature Sensor events. For instance, 1111b (-7) could be written to the TRIM_ADJ_OVER field and the Temperature Sensor event configured for an interrupt. When this interrupt occurs, the TRIM_ADJ_OVER bits could be updated to 0111b (+7), the interrupt cleared, and the Temperature Sensor event configured for a reset. In this way, a "warning" event (like an interrupt) could be detected as well as a "shutdown" event (at a higher temperature).

NOTE

Care must be taken to include a timeout during the clearing of the interrupt described in the example above. The Temperature Sensor has hysteresis and a timeout may be necessary to indicate if the interrupt flag has not cleared in a reasonable amount of time. If this is the case, then an error should be indicated.

Address: 0h base + 30Ch offset = 30Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS3IE	TS2IE	TS0IE	0	TRIM_ADJ_OVER[3:0]				0		TRIM_ADJ_UNDER[3:0]				DOUT_EN	AOUT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

PMCDIG_CTL_TD field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TS3IE	TS3IE Temperature Sensor input 3 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. **NOTE:** The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
17 TS2IE	TS2IE Temperature Sensor input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. **NOTE:** The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
18 TS0IE	TS0IE Temperature Sensor input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 TRIM_ADJ_ OVER[3:0]	Customer adjustable over trim register. These bits are a signed binary number that is added to the over temperature trim (NT_TD2)value. The TRIM value is protected from overflows and underflows due to this addition.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–29 TRIM_ADJ_ UNDER[3:0]	Customer adjustable trim register. These bits are a signed binary number that is added to the Under temperature trim value (NT_TD0). The TRIM value is protected from overflows and underflows due to this addition.
30 DOUT_EN	Digital Output Enable. This signal behavior has changed. The analog block now uses a 5 V signal for this input (and this field is not used) (see the Temperature Sensor Analog Chapter for details). 0 Disable 1 Enable
31 AOUT_EN	Analog Output Enable. 0 Disable 1 Enable

63.2.48 Temp Sensor FCCU Event Enable Register (PMCDIG_FEE_TD)

This Temp Sensor FCCU Event Enable Register controls whether the temperature sensor detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit (EPR_TD) is enabled, the temperature sensor detect assertion causes an FCCU event to be generated when the selected temperature exceeds the desired value.

Address: 0h base + 318h offset = 318h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FEE_TS3	FEE_TS2	0	FEE_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

PMCDIG_FEE_TD field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 FEE_TS3	FEE_TS3 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event.. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
29 FEE_TS2	FEE_TS2 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FEE_TS0	FEE_TS0 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.

63.2.49 Voltage Detect User Mode Test Register (PMCDIG_VD_UTST)

This register is used for testing of the LVDs and HVDs during user mode.

Before using the User Self-Test register field, the LVDs, HVDs, and PORs must be deasserted. If they are not deasserted, the self-test can cause an error. Check that all the LVDs, HVDs, and PORs are deasserted before starting this test.

Address: 0h base + 340h offset = 340h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				VOLT4_EN	VOLT3_EN	VOLT2_EN	VOLT1_EN	0				VOLT4	VOLT3	VOLT2	VOLT1
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								VD_UTST[5:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_VD_UTST field descriptions

Field	Description
0–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4 VOLT4_EN	VOLT4 enable. This bit defines whether internal voltage VOLT4 can generate a system reset. 0 Disabled. Only disable during Self Test process for VOLT4 1 Enabled. Normal resets are enabled.
5 VOLT3_EN	VOLT3 enable. This bit defines whether internal voltage VOLT4 can generate a system reset. 0 Disabled. Only disable during Self Test process for VOLT4 1 Enabled. Normal resets are enabled.
6 VOLT2_EN	VOLT2 enable. This bit defines whether internal voltage VOLT4 can generate a system reset. 0 Disabled. Only disable during Self Test process for VOLT4 1 Enabled. Normal resets are enabled.
7 VOLT1_EN	VOLT1 enable.

Table continues on the next page...

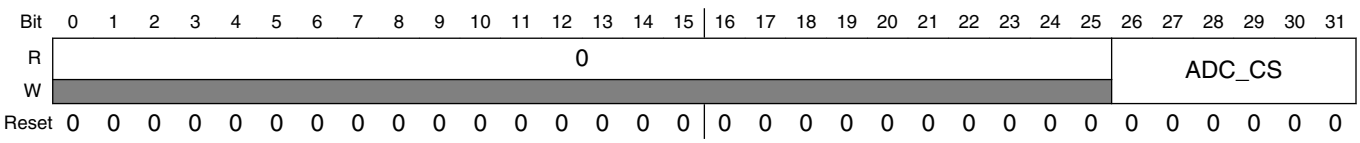
PMCDIG_VD_UTST field descriptions (continued)

Field	Description
	This bit defines whether internal voltage VOLT4 can generate a system reset. 0 Disabled. Only disable during Self Test process for VOLT4 1 Enabled. Normal resets are enabled.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 VOLT4	VOLT4 flag. This bit is the status flag associated with VOLT4. It is asserted asynchronously when the appropriate Self Test Mode bus is selected, and clears when a one is written to this bit location. 0 Currently no occurrence. 1 Voltage event occurrence detected.
13 VOLT3	VOLT3 flag. This bit is the status flag associated with VOLT3. It is asserted asynchronously when the appropriate Self Test Mode bus is selected, and clears when a one is written to this bit location. 0 Currently no occurrence. 1 Voltage event occurrence detected.
14 VOLT2	VOLT2 flag. This bit is the status flag associated with VOLT2. It is asserted asynchronously when the appropriate Self Test Mode bus is selected, and clears when a one is written to this bit location. 0 Currently no occurrence. 1 Voltage event occurrence detected.
15 VOLT1	VOLT1 flag. This bit is the status flag associated with VOLT1. It is asserted asynchronously when the appropriate Self Test Mode bus is selected, and clears when a one is written to this bit location. 0 Currently no occurrence. 1 Voltage event occurrence detected.
16–25 Reserved	This field is reserved.
26–31 VD_UTST[5:0]	Voltage Detect: User Test. User mode Test bits for testing of LVDs and HVDs.

63.2.50 ADC Channel Select Register (PMCDIG_ADC_CH)

This register configures the analog channel select mux of the PMC module.

Address: 0h base + 344h offset = 344h



PMCDIG_ADC_CH field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ADC_CS	ADC_CS Channel Select. These bits are used to control the analog PMC's test mux going to the ADC.

63.2.51 Voltage Regulator 1.2V Control Register (PMCDIG_VREG1P2_CTRL)

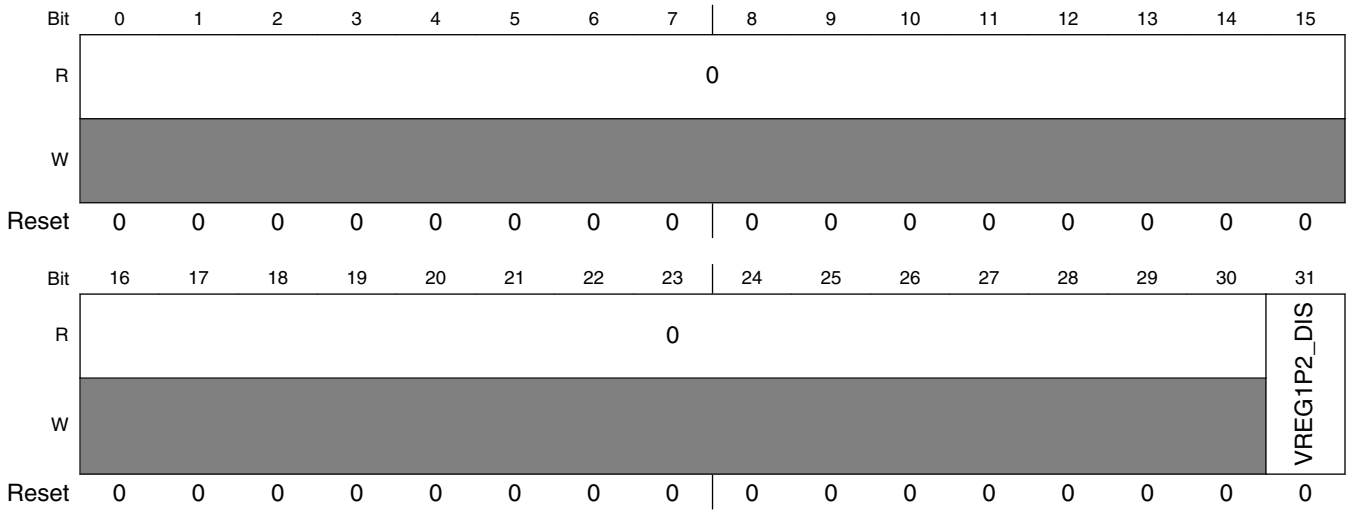
This register is used to disable the Auxiliary Regulator (also known as Spike or VREG1P2 Regulator) during an intentional power down event. The LSB of this register is a bit that is OR'd with the DCF control for the PMC_VREG1P2_ENB. If either the DCF loaded bit or the LSB of VREG1P2_CTRL is a 1, then the Auxiliary Regulator is disabled.

This bit is loaded during reset to match the value of the DCF location, and it is also only reset during a POR event.

This bit (VREG1P2_DIS) can only be used to turn off the Auxilliary Regulator, not turn it on.

This bit should be written high before any expected power down events.

Address: 0h base + 34Ch offset = 34Ch



PMCDIG_VREG1P2_CTRL field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 VREG1P2_DIS	This bit controls if the AUX Regulator (Vreg1p2) is disabled. During Power Down, this bit should be written to a 1 so that the AUX regulator is disabled. By default, the DCF loaded value (PMC_VREG1P2_ENB) is used to control the AUX regulator. 0 The enable of the VREG1P2 follows the DCF loaded bit. 1 VREG1P2 is disabled.

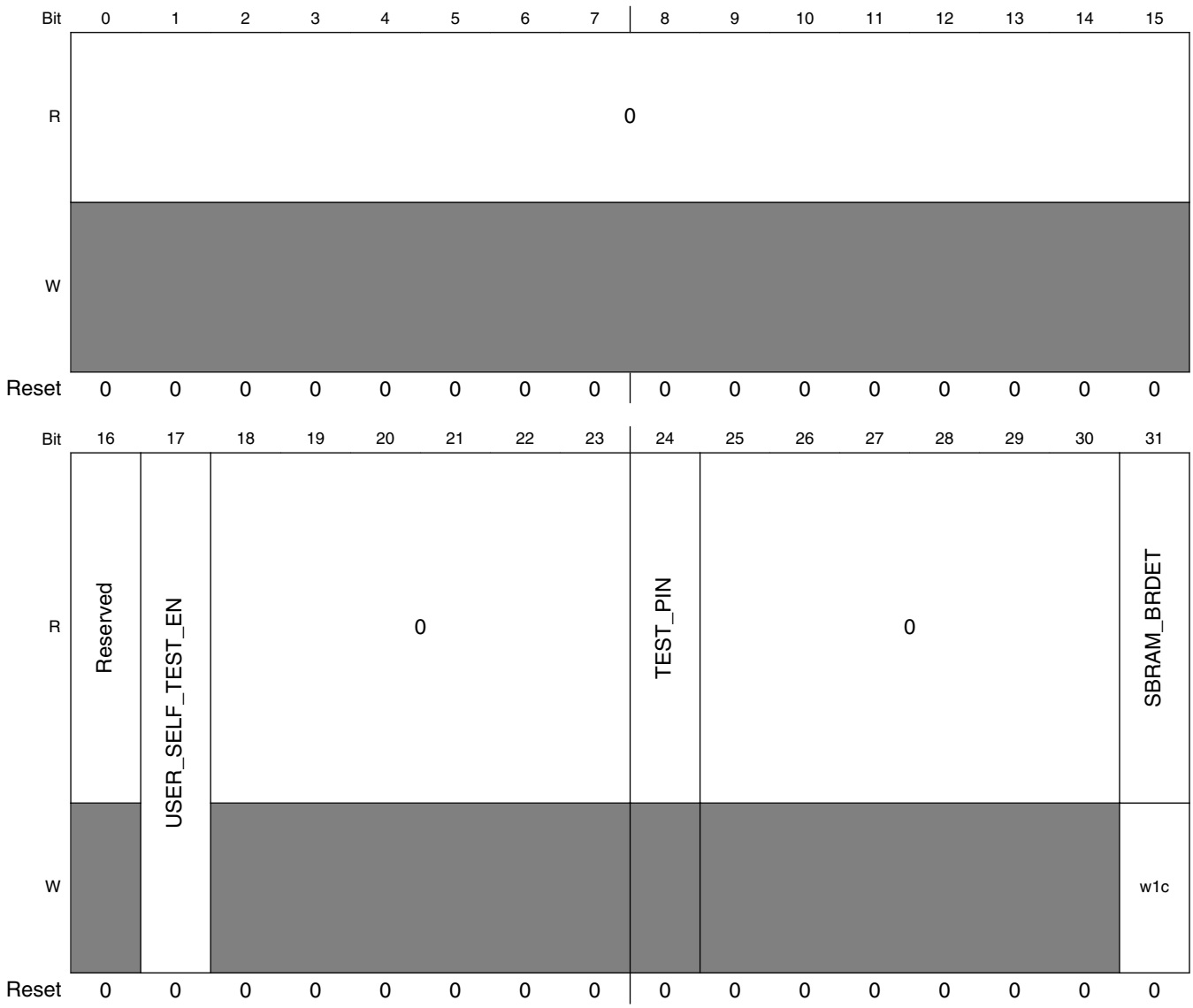
63.2.52 Module Control Register (PMCDIG_MCR)

This register contains several status and control bits. First, it the bit that is used to detect the status and to clear the Brown Out Detection logic for the Standby RAM module.

This register also contains the enable that is needed for the use of the PMC analog block User Self Test feature (USER_SELF_TEST_EN).

The proper procedure to enable these busses would be to write the value to the register containing the bus, then write this enable high, evaluate the test associated with this bus, and then write this bus enable bit low. After that, any further tests can be continued.

Address: 0h base + 350h offset = 350h



PMCDIG_MCR field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved.
17 USER_SELF_ TEST_EN	This bit allows the VD_UTST bits int he VD_UTST register to be enabled. This bit should be set after the VD_UTST bus has been updated, and cleared after the test has completed. The VD_UTST bus should not be updated while the USER_SELF_TEST_EN is enabled (high). 0 The VD_UTST bus is not enabled in the PMC analog. 1 The VD_UTST bus is enabled in the PMC analog.

Table continues on the next page...

PMCDIG_MCR field descriptions (continued)

Field	Description
18–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24 TEST_PIN	This bit is indicates the current value of the Test Pin. 0 Test Pin is not asserted. 1 Test Pin is asserted.
25–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
31 SBRAM_BRDET	SBRAM_BRDET. This bit is the detection status flag for the Standby RAM Brown Out Detection circuit. When a brown out event is detected in the Standby RAM logic, this signal asserts. In order to clear this flag, the brown out detection logic must no longer be detecting an event, and a one must be written to this bit location. 0 Currently no occurrence. 1 Standby RAM Brown Out occurrence detected.

63.3 Functional description

63.3.1 Analog PMC interface

The outputs from the PMC digital to the PMC analog circuit consists of trim registers for each LVD circuit, an enable for each LVD (tied to the enabled state), and ADC channel select controls. The ADC channel select controls are described in more detail in the ADC interface section of this chapter, and in the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

The signals from the PMC analog to the PMC digital logic consists of simply POR and LVD event indication signals. These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

There is a settling time for the analog voltage detect circuits specified at 5 µsec. The PMC digital counts clock edges after the loading of new flash trim values to verify that this time has passed. This time can be ≥ 5 µsec.

After a POR, some of the EPR registers may be set due to slow supply ramps or noisy supplies duirng power up. Clear these flags during initialization after a POR."

63.3.2 Temperature Sensor interface logic

The Temperature Sensor interface consists of two enables: an analog and digital enable, a set of mode registers, two 7-bit trim registers (sometimes referenced as one 14-bit trim register), and an overtemperature threshold select control.

The Temperature Sensor simply outputs three signals to the PMC digital logic: over_temp1, over_temp2, and under_temp signals. These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

There are two 4-bit signed registers (TRIM_ADJ_OVER[3:0] and TRIM_ADJ_UNDER[3:0]) that can be used by software to adjust the value of the Temp Sensor Trim values. The fourth bit of these registers indicates the sign.

TRIM_ADJ_OVER is added to TS2_TRIM and TRIM_ADJ_UNDER is added to TS0_TRIM. This addition is protected from overflow and underflow (the TRIM value will simply stay at max or min value).

Before changing either of the TRIM_ADJ registers, the Temp Sensor resets and interrupts should be disabled via the TEMP bits in the PMCDIG_REE_TD register and TSIE bits in the PMCDIG_CTL_TD register. Resets and interrupts for the Temp Sensor should not be enabled until the Temp Sensor Status bits (TS bits in GR_S register) indicate that there is no longer a temperature event occurring.

For more information regarding the Temperature Sensor function, please refer to the Temperature Sensor chapter in this Reference Manual chapter.

NOTE

When the temperature exceeds the defined threshold, the bits of PMCDIG_EPR_TD register are set. These bits are cleared when a "1" is written to the bit location. The current status of the Temperature Status can be determined with PMCDIG_GR_S[TS3], PMCDIG_GR_S[TS2] and PMCDIG_GR_S[TS0] bits of the PMCDIG_GR_S register. These bits also indicate if the temperature threshold has been exceeded previously.

63.3.3 Standby RAM regulator interface logic

The Standby RAM Regulator interface consists of one bit that indicates that a Standby RAM Brown-out event has occurred, and a method to clear this indication (by writing a one to the flag bit).

For more information regarding the Standby RAM Regulator function, please refer to the Standby RAM Regulator Analog Chapter.

63.3.4 Power On Reset (MC_RGM phase gates)

The PMC digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMC and Temperature Sensor Analog blocks.

There are two signals for each the PMC and Temp Sensor blocks; one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMC digital logic also indicates when is OK to for the RGM to allow transition to the next POR phase.

The digital PMC also uses signals from the RGM to indicate when some of the reset signals should be masked during the POR sequence.

During the POR phase 3 exit, the PMC digital logic uses all the enabled Voltage Detects to determine if it is correct to exit from Phase 3. Once all the enabled HVD/LVDs have become deasserted and the time for the LVD/HVD circuits to adjust to new trim values (Tlvdtrim (and other system time counts): see PMC analog block guide) has elapsed, the signal is then sent to the RGM that the correct voltage levels have been achieved to exit from Phase 3.

There are nine POR signals:

- POR_VDDREG
- POR_VDDC
- POR081_C
- POR085_C
- POR085_F
- POR098_C
- POR098_F
- POR250_F
- POR260_C

63.3.5 Flash memory interface (DCF client usage)

The PMC digital is updated during POR from flash memory via a DCF client. This load is used for several pieces of data, but most of the data is trim values being used for analog modules (PMC, Temperature Sensor, and ADC Bandgap).

The trim bits that are loaded from flash memory are not readable. These trim values loaded via the DCF client are compacted (they are written several at a time).

A DCF client is used to interface to flash memory for these early data accesses. This client also performs triple voting to keep the data safe from stray bit flips. If any issues occur during any of the DCF data loads (like a second write of different data), the DCF client will indicate an error to the FCCU.

Another set of data that is loaded during the DCF single read process are updated values for the REE and RES registers.

If the REE bit is loaded via the DCF as set (enabled), then it is not possible to clear this REE bit. The reset values of the REE DCF bits are all low (thus to allow the bits to be rewriteable), but the reset value of the REE register fields are high (to enable the resets normally out of POR).

The REE DCF bits loaded are compacted into a single DCF write (they do not match the IPS Bus register bit locations). The REE DCF loaded bits do not have the same limitation that they can only be written once and any further writes must be to the same value. The REE DCF bits can be rewritten to a new value during a new reset event.

When the REE DCF bits are updated, the REE and RES register bits are updated at the same time to the new value.

All of the DCF clients are reset via the destructive reset signal (see the "Device Configuration Format (DCF) Records" chapter for details).

Trim data is read from the flash memory's differential read space (slower access, but less issues with voltage variances). Once the differential reads are completed, the flash memory indicates read completion to PMC_dig via the SSCM module (see the flash memory chapter for details on differential reads and required voltage levels).

Once the PMC trim values have been read, the analog circuit has been given time to update to the new trim values, and the LVDs have all deasserted, a signal is asserted to the flash memory to indicate that the voltage is high enough for faster single access reads.

All of the LVD/HVD signals must indicate that a valid voltage has been reached before the voltage indication will be sent to the flash memory.

63.3.6 Other module interfaces

The FCCU block receives several signals to indicate a fault has occurred. The first is a safety error from the DCF interface indicating that some error has happened during the load of the trim data from the flash during POR. All of the DCF client safety error signals are combined in the PMC digital logic and then sent to the FCCU.

The second set of signals indicate that an HVD, LVD, or temperature event has occurred.

- The LVDs that are sent to the FCCU are:
  - LVD108_C
  - LVD108_F
  - LVD108_P
  - LVD112_C
  - LVD270_C
  - LVD270_F
  - LVD270_IF
  - LVD270_IJ
  - LVD270_IM
  - LVD270_O
  - LVD295_F
  - LVD295_A
  - LVD360_IM
  - LVD400_IM
  - LVD400_A
- The HVDs that are sent to the FCCU are:
  - HVD140_C
  - HVD145_F
  - HVD145_C
  - HVD360_F
  - HVD600_C
  - HVD600_A

The Temp Sensor outputs tmpsns3, tmpsns2, and tmpsns0 are all combined and sent to the FCCU.

For more information regarding the FCCU block and these signals, please refer to the FCCU chapter.

The HSM module receives four signals from the PMC digital. There are two signals that are sent to the HSM that includes both over_temp1 and over_temp2 values. The under_temp signal is also sent to the HSM as well. The last signal indicates that the PMC has detected a voltage higher than 1.4V on the internal cold detection point.

For more information regarding the HSM block and these signals, please refer to the HSM chapter of the MPC5700 Family Microcontroller Security Reference Manual.

Chapter 64
Power Control Unit (MC_PCU)

64.1 Introduction

64.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMC peripheral to the MC_PCU address space.

The following figure depicts the MC_PCU block diagram.

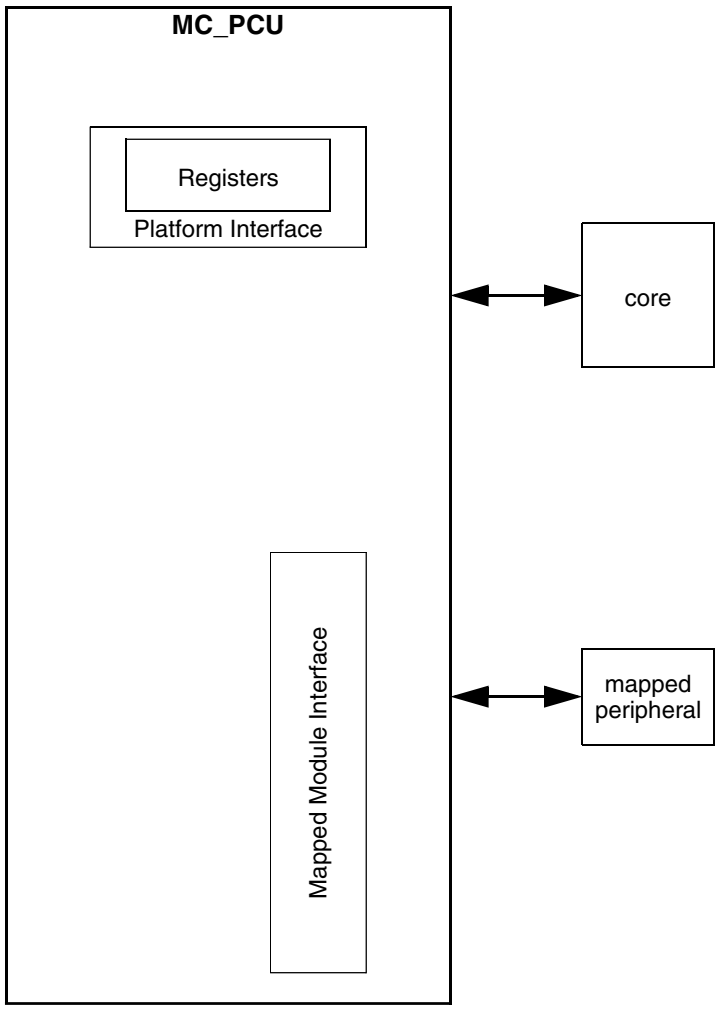


Figure 64-1. MC_PCU block diagram

64.1.2 Features

The MC_PCU includes the following features:

- maps the PMC registers to the MC_PCU address space

64.2 Memory Map and Register Definition

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at address 0x0040, as a half-word at address 0x0042, or as a byte at address 0x0043.

NOTE

All registers will be accessible in "User mode" provided the access to this mode is enabled through REGPROT settings. Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error(only if SSCM_ERROR[RAE] is set)

MC_PCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
40	Power Domain Status Register (MC_PCU_PSTAT)	32	R	0000_0001h	64.2.1/3425

64.2.1 Power Domain Status Register (MC_PCU_PSTAT)

This register reflects the power status of all available power domains.

Address: FFFA_0000h base + 40h offset = FFFA_0040h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															PD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_PCU_PSTAT field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PD0	Power status for power domain # *n* 0 Power domain is inoperable 1 Power domain is operable

MC_PCU_PSTAT field descriptions (continued)

Field	Description

64.3 Functional Description

Please see the PMC_dig chapter for specific details on the PMC control registers.

Chapter 65
Mode Entry Module (MC_ME)

65.1 Introduction

65.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

The following figure shows the MC_ME block diagram.

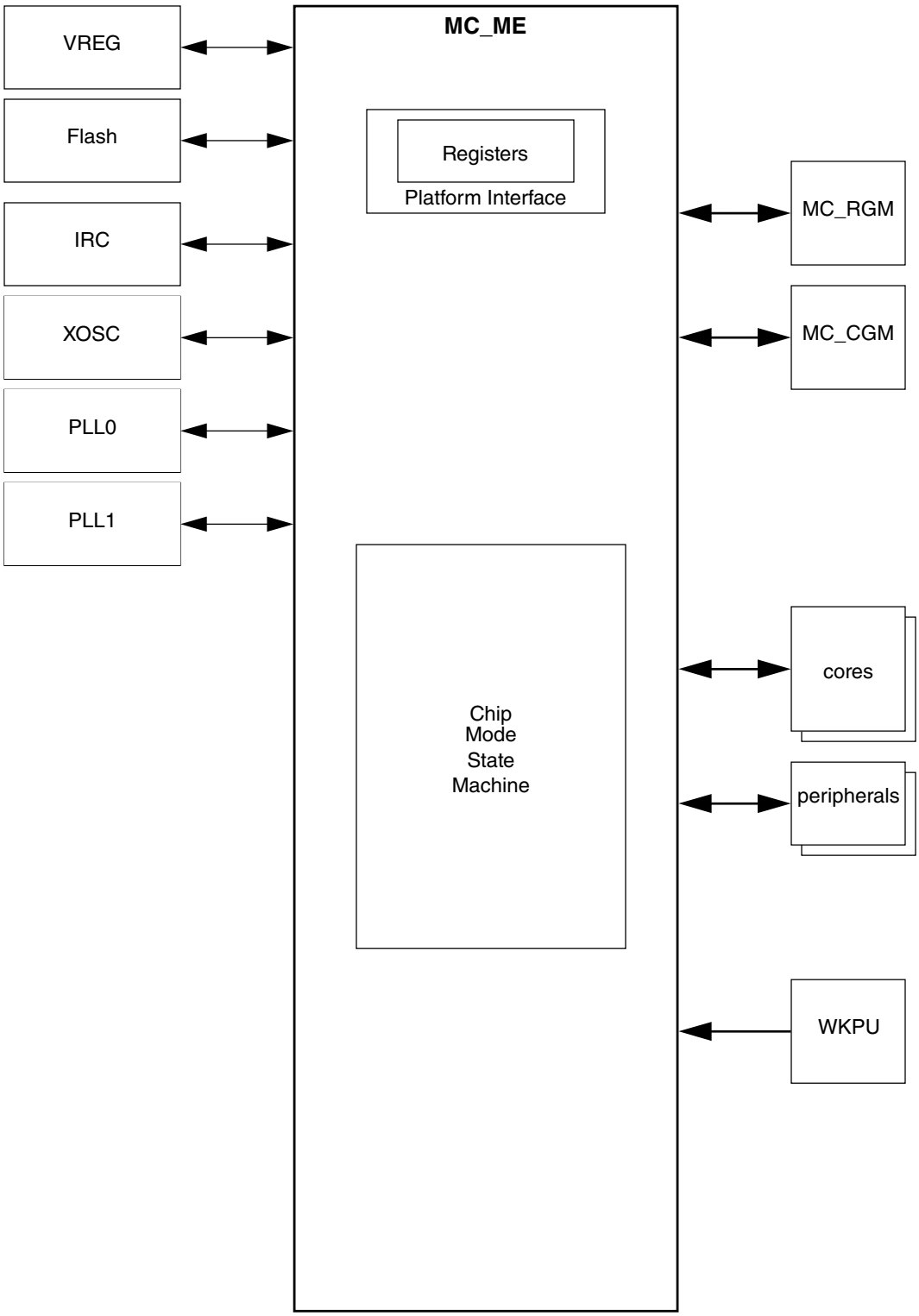


Figure 65-1. MC_ME Block Diagram

65.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers
- additional core clock gating and boot address control based on the ME_CCTLn and ME_CADDRn registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

65.1.3 Modes of operation

- The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0…3, HALT0, and STOP0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0…3 are the chip software running modes.

The following table describes the MC_ME modes.

Table 65-1. MC_ME Mode Descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAF when present is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0…3	system reset assertion, RUN0…3, TEST via software, SAFE via software or hardware failure.
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0…3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0…3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0…3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0…3 modes, HALT0, STOP0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event[1] or wakeup event

1. If the user does not want the part to be woken up from certain Interrupt events, then the user needs to explicitly disable those interrupts in the respective peripherals. Disabling the respective interrupt channels in the Interrupt Controller will not suffice.

65.2 External signal description

The MC_ME has no connections to any external pins.

65.3 Memory map and register definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

NOTE

All registers will be accessible in "User mode" provided the access to this mode is enabled through REGPROT settings. Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error(only if SSCM_ERROR[RAE] is set)

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

MC_ME memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Status Register (MC_ME_GS)	32	R	0C13_0010h	65.3.1/3436
4	Mode Control Register (MC_ME_MCTL)	32	R/W	3000_A50Fh	65.3.2/3439
8	Mode Enable Register (MC_ME_ME)	32	R/W	0000_801Dh	65.3.3/3441
C	Interrupt Status Register (MC_ME_IS)	32	w1c	0000_0000h	65.3.4/3443
10	Interrupt Mask Register (MC_ME_IM)	32	R/W	0000_0000h	65.3.5/3444
14	Invalid Mode Transition Status Register (MC_ME_IMTS)	32	w1c	0000_0000h	65.3.6/3446
18	Debug Mode Transition Status Register (MC_ME_DMTS)	32	R	0000_0000h	65.3.7/3447
20	RESET Mode Configuration Register (MC_ME_RESET_MC)	32	R	0013_0010h	65.3.8/3452
24	TEST Mode Configuration Register (MC_ME_TEST_MC)	32	R/W	0013_0010h	65.3.9/3455

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	SAFE Mode Configuration Register (MC_ME_SAFE_MC)	32	R/W	0093_0010h	65.3.10/ 3457
2C	DRUN Mode Configuration Register (MC_ME_DRUN_MC)	32	R/W	0013_0010h	65.3.11/ 3459
30	RUN0 Mode Configuration Register (MC_ME_RUN0_MC)	32	R/W	0013_0010h	65.3.12/ 3462
34	RUN1 Mode Configuration Register (MC_ME_RUN1_MC)	32	R/W	0013_0010h	65.3.13/ 3464
38	RUN2 Mode Configuration Register (MC_ME_RUN2_MC)	32	R/W	0013_0010h	65.3.14/ 3467
3C	RUN3 Mode Configuration Register (MC_ME_RUN3_MC)	32	R/W	0013_0010h	65.3.15/ 3469
40	HALT0 Mode Configuration Register (MC_ME_HALT0_MC)	32	R/W	0012_0010h	65.3.16/ 3471
48	STOP0 Mode Configuration Register (MC_ME_STOP0_MC)	32	R/W	0011_0010h	65.3.17/ 3474
60	Peripheral Status Register 0 (MC_ME_PS0)	32	R	0000_0000h	65.3.18/ 3477
64	Peripheral Status Register 1 (MC_ME_PS1)	32	R	0000_0000h	65.3.19/ 3479
68	Peripheral Status Register 2 (MC_ME_PS2)	32	R	0000_0000h	65.3.20/ 3481
6C	Peripheral Status Register 3 (MC_ME_PS3)	32	R	0000_0000h	65.3.21/ 3484
70	Peripheral Status Register 4 (MC_ME_PS4)	32	R	0000_0000h	65.3.22/ 3487
74	Peripheral Status Register 5 (MC_ME_PS5)	32	R	0000_0000h	65.3.23/ 3488
78	Peripheral Status Register 6 (MC_ME_PS6)	32	R	0000_0000h	65.3.24/ 3490
7C	Peripheral Status Register 7 (MC_ME_PS7)	32	R	0000_0000h	65.3.25/ 3492
80	Run Peripheral Configuration Register (MC_ME_RUN_PC0)	32	R/W	0000_0000h	65.3.26/ 3495
84	Run Peripheral Configuration Register (MC_ME_RUN_PC1)	32	R/W	0000_0000h	65.3.26/ 3495
88	Run Peripheral Configuration Register (MC_ME_RUN_PC2)	32	R/W	0000_0000h	65.3.26/ 3495
8C	Run Peripheral Configuration Register (MC_ME_RUN_PC3)	32	R/W	0000_0000h	65.3.26/ 3495
90	Run Peripheral Configuration Register (MC_ME_RUN_PC4)	32	R/W	0000_0000h	65.3.26/ 3495
94	Run Peripheral Configuration Register (MC_ME_RUN_PC5)	32	R/W	0000_0000h	65.3.26/ 3495

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
98	Run Peripheral Configuration Register (MC_ME_RUN_PC6)	32	R/W	0000_0000h	65.3.26/ 3495
9C	Run Peripheral Configuration Register (MC_ME_RUN_PC7)	32	R/W	0000_0000h	65.3.26/ 3495
A0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC0)	32	R/W	0000_0000h	65.3.27/ 3496
A4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC1)	32	R/W	0000_0000h	65.3.27/ 3496
A8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC2)	32	R/W	0000_0000h	65.3.27/ 3496
AC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC3)	32	R/W	0000_0000h	65.3.27/ 3496
B0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC4)	32	R/W	0000_0000h	65.3.27/ 3496
B4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC5)	32	R/W	0000_0000h	65.3.27/ 3496
B8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC6)	32	R/W	0000_0000h	65.3.27/ 3496
BC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC7)	32	R/W	0000_0000h	65.3.27/ 3496
C3	EBI_0 Peripheral Control Register (MC_ME_PCTL3)	8	R/W	00h	65.3.28/ 3497
C9	LFAST_0 Peripheral Control Register (MC_ME_PCTL9)	8	R/W	00h	65.3.29/ 3498
CB	SIPI_0 Peripheral Control Register (MC_ME_PCTL11)	8	R/W	00h	65.3.30/ 3499
CF	SIUL Peripheral Control Register (MC_ME_PCTL15)	8	R/W	00h	65.3.31/ 3500
DE	PIT_RTC_0 Peripheral Control Register (MC_ME_PCTL30)	8	R/W	00h	65.3.32/ 3501
DF	PIT_RTC_1 Peripheral Control Register (MC_ME_PCTL31)	8	R/W	00h	65.3.33/ 3502
E4	DMAMUX_0 Peripheral Control Register (MC_ME_PCTL36)	8	R/W	00h	65.3.34/ 3503
E6	CRC_0 Peripheral Control Register (MC_ME_PCTL38)	8	R/W	00h	65.3.35/ 3504
F8	ADCSD_8 Peripheral Control Register (MC_ME_PCTL56)	8	R/W	00h	65.3.36/ 3505
F9	ADCSD_6 Peripheral Control Register (MC_ME_PCTL57)	8	R/W	00h	65.3.37/ 3506
FA	ADCSD_4 Peripheral Control Register (MC_ME_PCTL58)	8	R/W	00h	65.3.38/ 3507
FB	ADCSD_2 Peripheral Control Register (MC_ME_PCTL59)	8	R/W	00h	65.3.39/ 3508

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FC	ADCSD_0 Peripheral Control Register (MC_ME_PCTL60)	8	R/W	00h	65.3.40/ 3509
103	MCAN_4 Peripheral Control Register (MC_ME_PCTL67)	8	R/W	00h	65.3.41/ 3510
104	MCAN_3 Peripheral Control Register (MC_ME_PCTL68)	8	R/W	00h	65.3.42/ 3511
105	MCAN_2 Peripheral Control Register (MC_ME_PCTL69)	8	R/W	00h	65.3.43/ 3512
106	MCAN_1 Peripheral Control Register (MC_ME_PCTL70)	8	R/W	00h	65.3.44/ 3513
108	TTCAN Peripheral Control Register (MC_ME_PCTL72)	8	R/W	00h	65.3.45/ 3514
10A	CAN_RAM_CTRL Peripheral Control Register (MC_ME_PCTL74)	8	R/W	00h	65.3.46/ 3515
114	LINFlexD_16 Peripheral Control Register (MC_ME_PCTL84)	8	R/W	00h	65.3.47/ 3516
115	LINFlexD_14 Peripheral Control Register (MC_ME_PCTL85)	8	R/W	00h	65.3.48/ 3517
11B	LINFlexD_1 Peripheral Control Register (MC_ME_PCTL91)	8	R/W	00h	65.3.49/ 3518
11C	LINFlexD_0 Peripheral Control Register (MC_ME_PCTL92)	8	R/W	00h	65.3.50/ 3519
11D	DSPI_12 Peripheral Control Register (MC_ME_PCTL93)	8	R/W	00h	65.3.51/ 3520
120	DSPI_6 Peripheral Control Register (MC_ME_PCTL96)	8	R/W	00h	65.3.52/ 3521
121	DSPI_4 Peripheral Control Register (MC_ME_PCTL97)	8	R/W	00h	65.3.53/ 3522
122	DSPI_1 Peripheral Control Register (MC_ME_PCTL98)	8	R/W	00h	65.3.54/ 3523
123	DSPI_0 Peripheral Control Register (MC_ME_PCTL99)	8	R/W	00h	65.3.55/ 3524
125	IIC_0 Peripheral Control Register (MC_ME_PCTL101)	8	R/W	00h	65.3.56/ 3525
128	SENT_0 Peripheral Control Register (MC_ME_PCTL104)	8	R/W	00h	65.3.57/ 3526
12B	FLEXRAY_0 Peripheral Control Register (MC_ME_PCTL107)	8	R/W	00h	65.3.58/ 3527
12F	PSI5_0 Peripheral Control Register (MC_ME_PCTL111)	8	R/W	00h	65.3.59/ 3528
130	ADCSAR_b Peripheral Control Register (MC_ME_PCTL112)	8	R/W	00h	65.3.60/ 3529
13B	ADCSAR_4 Peripheral Control Register (MC_ME_PCTL123)	8	R/W	00h	65.3.61/ 3530

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13F	ADCSAR_0 Peripheral Control Register (MC_ME_PCTL127)	8	R/W	00h	65.3.62/ 3531
140	GTMINT Peripheral Control Register (MC_ME_PCTL128)	8	R/W	00h	65.3.63/ 3532
162	PSI5_S_0 Peripheral Control Register (MC_ME_PCTL162)	8	R/W	00h	65.3.64/ 3533
166	CRC_1 Peripheral Control Register (MC_ME_PCTL166)	8	R/W	00h	65.3.65/ 3534
178	ADCSD_9 Peripheral Control Register (MC_ME_PCTL184)	8	R/W	00h	65.3.66/ 3535
179	ADCSD_7 Peripheral Control Register (MC_ME_PCTL185)	8	R/W	00h	65.3.67/ 3536
17A	ADCSD_5 Peripheral Control Register (MC_ME_PCTL186)	8	R/W	00h	65.3.68/ 3537
17B	ADCSD_3 Peripheral Control Register (MC_ME_PCTL187)	8	R/W	00h	65.3.69/ 3538
17C	ADCSD_1 Peripheral Control Register (MC_ME_PCTL188)	8	R/W	00h	65.3.70/ 3539
195	LINFlexD_15 Peripheral Control Register (MC_ME_PCTL213)	8	R/W	00h	65.3.71/ 3540
19C	LINFlexD_2 Peripheral Control Register (MC_ME_PCTL220)	8	R/W	00h	65.3.72/ 3541
1A1	DSPI_5 Peripheral Control Register (MC_ME_PCTL225)	8	R/W	00h	65.3.73/ 3542
1A2	DSPI_3 Peripheral Control Register (MC_ME_PCTL226)	8	R/W	00h	65.3.74/ 3543
1A3	DSPI_2 Peripheral Control Register (MC_ME_PCTL227)	8	R/W	00h	65.3.75/ 3544
1A5	IIC_1 Peripheral Control Register (MC_ME_PCTL229)	8	R/W	00h	65.3.76/ 3545
1A8	SENT_1 Peripheral Control Register (MC_ME_PCTL232)	8	R/W	00h	65.3.77/ 3546
1AB	FLEXRAY_1 Peripheral Control Register (MC_ME_PCTL235)	8	R/W	00h	65.3.78/ 3547
1AF	PSI5_1 Peripheral Control Register (MC_ME_PCTL239)	8	R/W	00h	65.3.79/ 3548
1B5	ADCSAR_10 Peripheral Control Register (MC_ME_PCTL245)	8	R/W	00h	65.3.80/ 3549
1B6	ADCSAR_9 Peripheral Control Register (MC_ME_PCTL246)	8	R/W	00h	65.3.81/ 3550
1B7	ADCSAR_8 Peripheral Control Register (MC_ME_PCTL247)	8	R/W	00h	65.3.82/ 3551
1B8	ADCSAR_7 Peripheral Control Register (MC_ME_PCTL248)	8	R/W	00h	65.3.83/ 3551

Table continues on the next page...

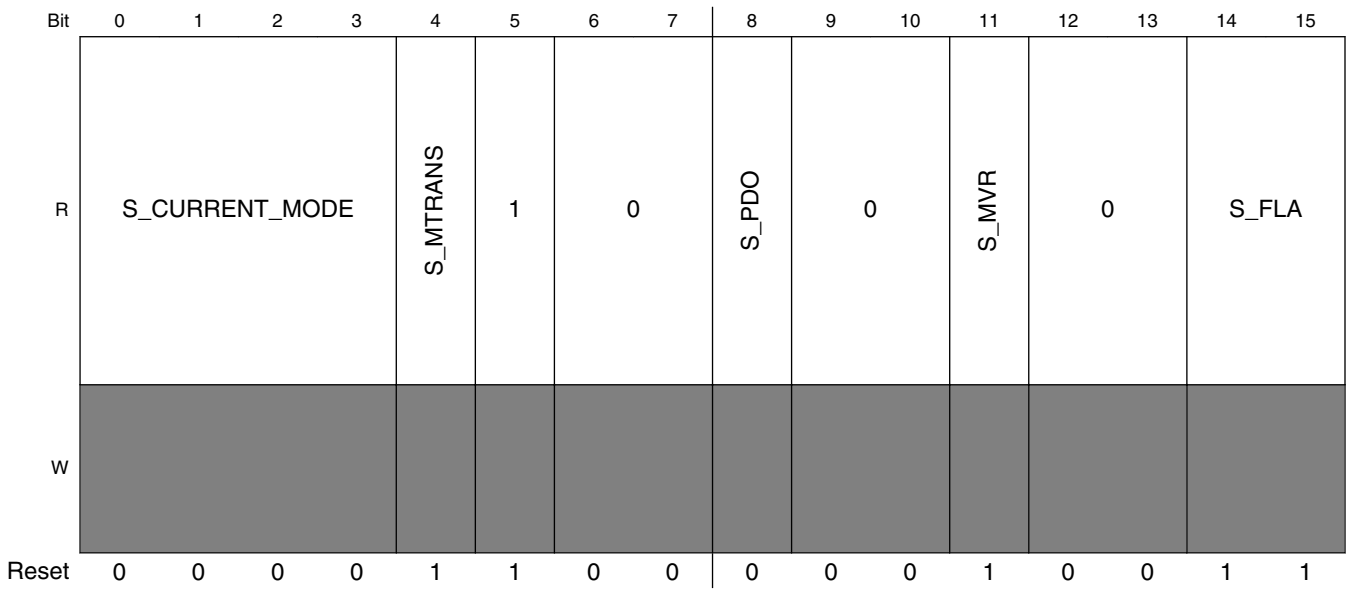
MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1B9	ADCSAR_6 Peripheral Control Register (MC_ME_PCTL249)	8	R/W	00h	65.3.84/ 3553
1BA	ADCSAR_5 Peripheral Control Register (MC_ME_PCTL250)	8	R/W	00h	65.3.85/ 3554
1BC	ADCSAR_3 Peripheral Control Register (MC_ME_PCTL252)	8	R/W	00h	65.3.86/ 3555
1BD	ADCSAR_2 Peripheral Control Register (MC_ME_PCTL253)	8	R/W	00h	65.3.87/ 3556
1BE	ADCSAR_1 Peripheral Control Register (MC_ME_PCTL254)	8	R/W	00h	65.3.88/ 3557
1C0	Core Status Register (MC_ME_CS)	32	R	0000_0000h	65.3.89/ 3558
1C4	CORE0 Core Control Register (MC_ME_CCTL0)	16	R/W	00FEh	65.3.90/ 3559
1C6	CORE1 Core Control Register (MC_ME_CCTL1)	16	R/W	0000h	65.3.91/ 3560
1C8	CORE2 Core Control Register (MC_ME_CCTL2)	16	R/W	0000h	65.3.92/ 3562
1CA	CORE3 Core Control Register (MC_ME_CCTL3)	16	R/W	0000h	65.3.93/ 3563
1CC	CORE4 Core Control Register (MC_ME_CCTL4)	16	R/W	0000h	65.3.94/ 3565
1E0	CORE0 Core Address Register (MC_ME_CADDR0)	32	R/W	See section	65.3.95/ 3566
1E4	CORE1 Core Address Register (MC_ME_CADDR1)	32	R/W	0000_0000h	65.3.96/ 3567
1E8	CORE2 Core Address Register (MC_ME_CADDR2)	32	R/W	0000_0000h	65.3.97/ 3568
1EC	CORE3 Core Address Register (MC_ME_CADDR3)	32	R/W	0000_0000h	65.3.98/ 3569
1F0	CORE4 Core Address Register (MC_ME_CADDR4)	32	R/W	See section	65.3.99/ 3569

65.3.1 Global Status Register (MC_ME_GS)

This register contains global mode status.

Address: 0h base + 0h offset = 0h



MC_ME_GS field descriptions

Field	Description
0–3 S_CURRENT_ MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 S_PDO	Output power-down status - This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In STOP0 mode, the slew rate control mechanism of the output pads is disabled to reduce power consumption, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode and available for use
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 S_PLL1	secondary PLL status 0 secondary PLL is not stable 1 secondary PLL is providing a stable clock
25 S_PLL0	primary PLL status 0 primary PLL is not stable 1 primary PLL is providing a stable clock
26 S_XOSC	external crystal oscillator status

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	0 external crystal oscillator is not stable 1 external crystal oscillator is providing a stable clock
27 S_IRC	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable 1 16 MHz internal RC oscillator is providing a stable clock
28–31 S_SYSCLK	System clock switch status - These bits specify the system clock currently used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

65.3.2 Mode Control Register (MC_ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

NOTE

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TARGET_MODE				0																											
W																	KEY															
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

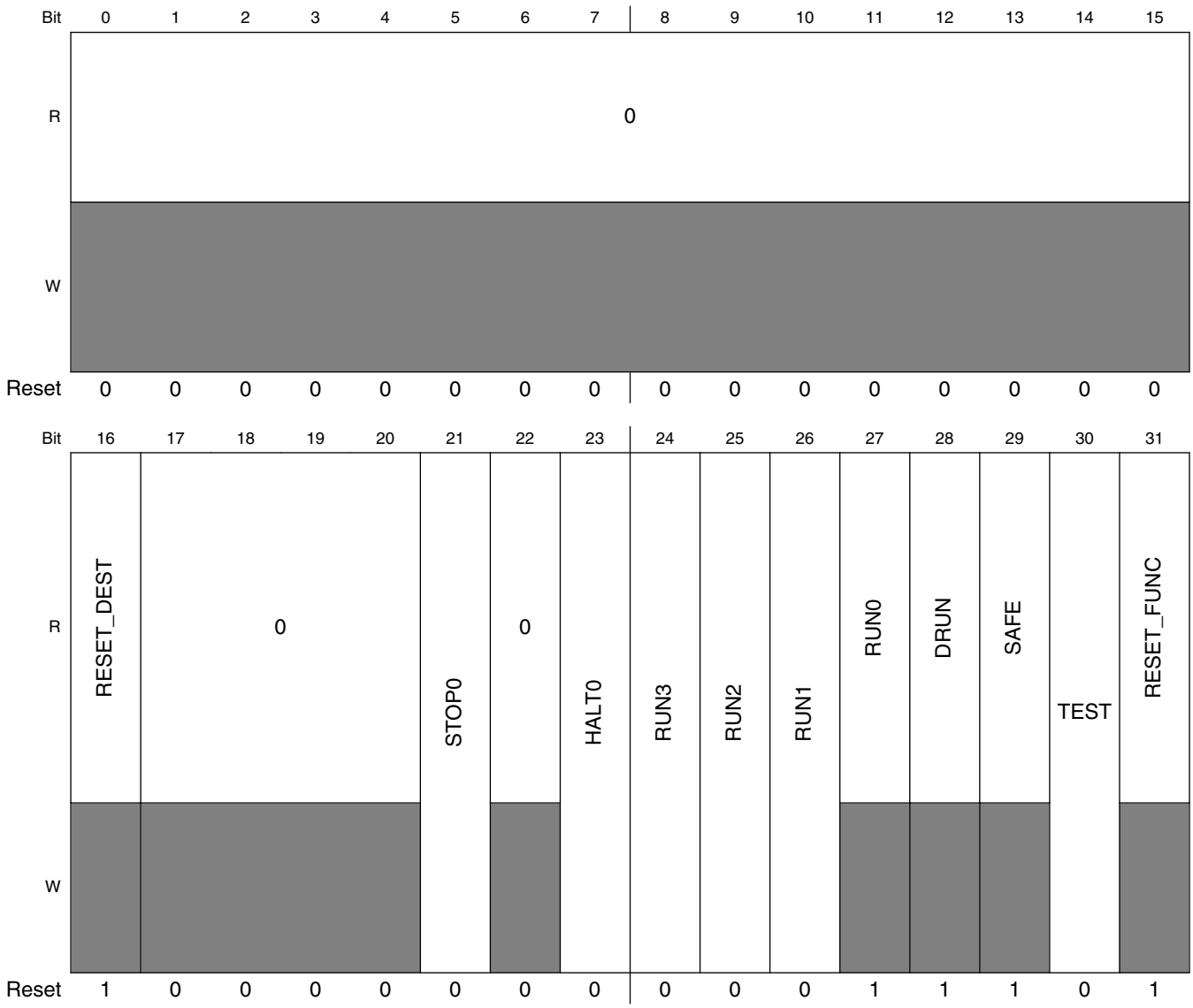
MC_ME_MCTL field descriptions

Field	Description
0–3 TARGET_MODE	Target chip mode - These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0…3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 RESET (triggers a 'destructive' reset event)
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 KEY	Control key - These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

65.3.3 Mode Enable Register (MC_ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET_FUNC, SAFE, DRUN, RUN0, and RESET_DEST modes are always enabled.

Address: 0h base + 8h offset = 8h



MC_ME_ME field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

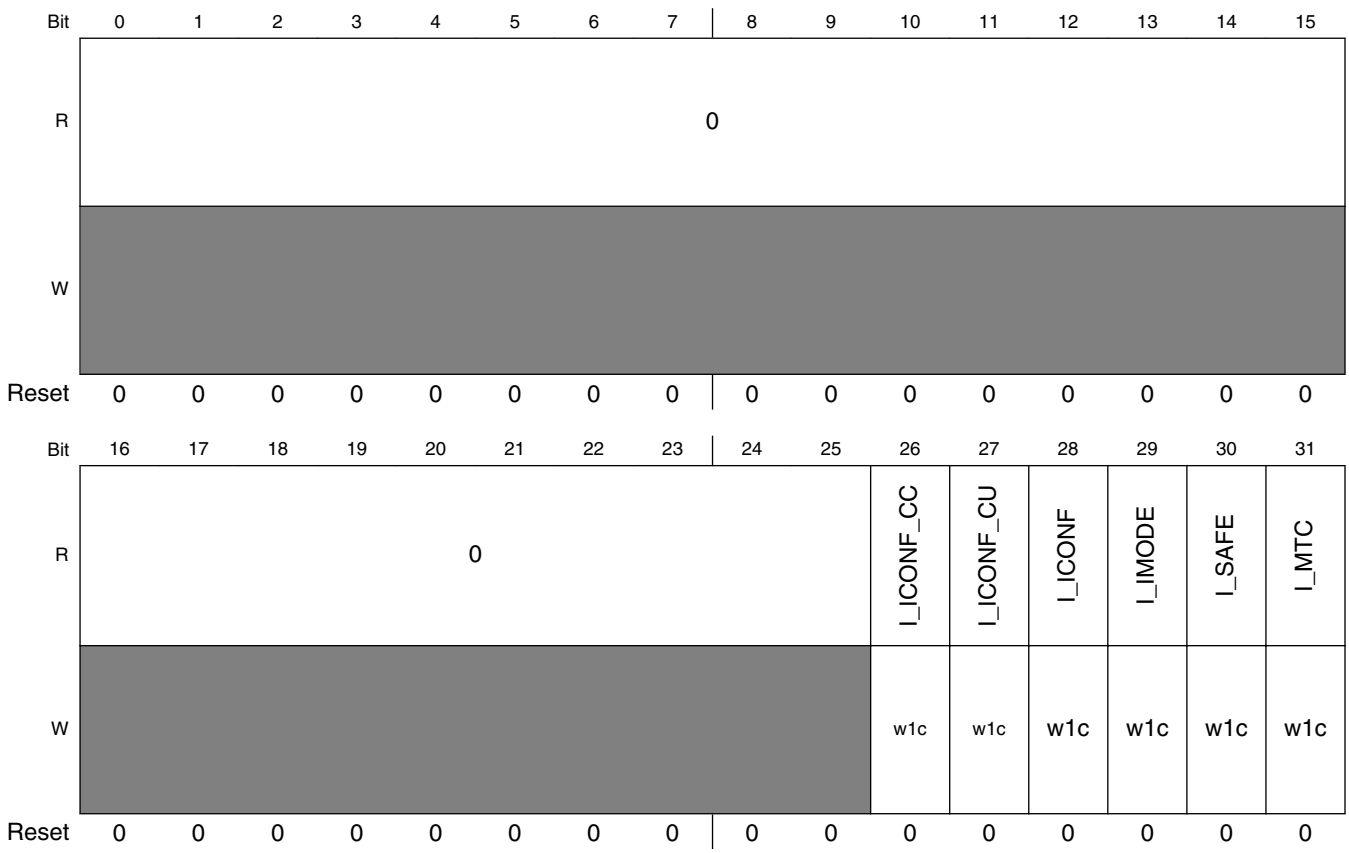
MC_ME_ME field descriptions (continued)

Field	Description
16 RESET_DEST	'destructive' RESET mode enable 1 'destructive' RESET mode is enabled
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled
30 TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
31 RESET_FUNC	'functional' RESET mode enable 1 'functional' RESET mode is enabled

65.3.4 Interrupt Status Register (MC_ME_IS)

This register provides the current interrupt status.

Address: 0h base + Ch offset = Ch



MC_ME_IS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) - This bit is set if a write access to one of the ME_CCTLn registers is attempted while a mode transition is in progress. 0 No write to an ME_CCTLn register was attempted during an ongoing mode transition 1 A write to an ME_CCTLn register was attempted during an ongoing mode transition
27 I_ICONF_CU	Invalid mode configuration interrupt (Clock Usage) - This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending

Table continues on the next page...

MC_ME_IS field descriptions (continued)

Field	Description
28 I_ICONF	Invalid mode configuration interrupt - This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt - This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt - This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt - This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

65.3.5 Interrupt Mask Register (MC_ME_IM)

This register controls whether an event generates an interrupt or not.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

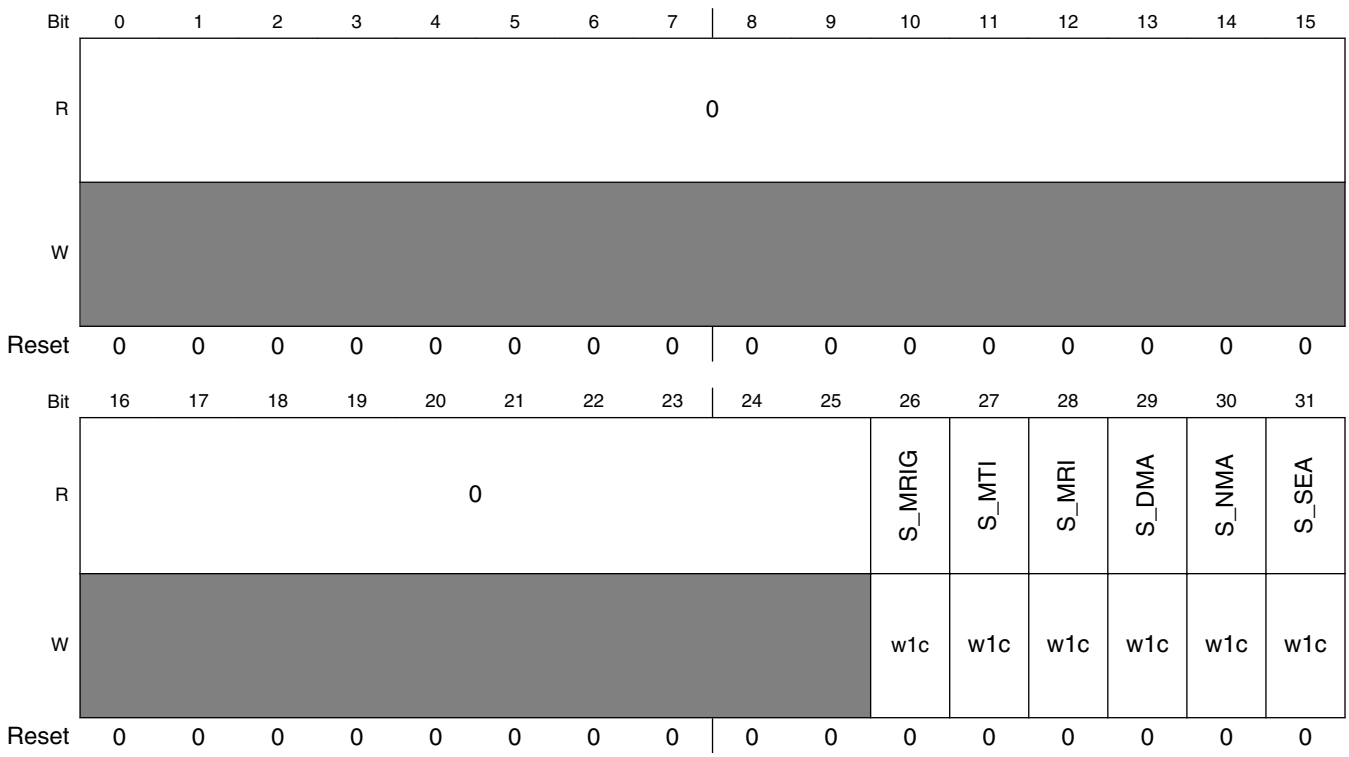
MC_ME_IM field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

65.3.6 Invalid Mode Transition Status Register (MC_ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Address: 0h base + 14h offset = 14h



MC_ME_IMTS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_MRIG	Mode Request Ignored Status - This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress. This bit is also set if a transition to STOP/HALT mode is requested when a wakeup is active. It is cleared by writing 1 to this bit. 0 Mode transition requested is not ignored 1 Mode transition requested is ignored
27 S_MTI	Mode Transition Illegal status - This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is '1'). Please refer to Mode transition interrupts for the exceptions to this behavior. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
28 S_MRI	Mode Request Illegal status - This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit.

Table continues on the next page...

MC_ME_IMTS field descriptions (continued)

Field	Description
	0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	Disabled Mode Access status - This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	Non-existing Mode Access status - This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status - This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

65.3.7 Debug Mode Transition Status Register (MC_ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected.

NOTE

The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0				MPH_BUSY	0		PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	0	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_DMTS field descriptions

Field	Description
0–3 PREVIOUS_ MODE	Previous chip mode - These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator - This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has power-up/down sequencing. 0 Handshake is not busy 1 Handshake is busy
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PMC_PROG	MC_PCU Mode Change in Progress indicator - This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator - This bit reads 0.
13 CCKL_PROG	Core Clock Enable/Disable in Progress - This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress - This bit is set while the progressive system clock switching process is in progress, or the peripheral clock switching is ongoing, or core clock switching is ongoing, or a power sequence process is ongoing (provided the target mode is not STANDB0). 0 PCS is not in progress 1 PCS is in progress
15 SMR	SAFE mode request from MC_RGM is active indicator - This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
	0 A SAFE mode request is not active 1 A SAFE mode request is active
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator - This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator - This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
19 IRC_SC	IRC State Change during mode transition indicator - This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator - This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A 'secondary clock source' is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status - 0 No system clock source switching is pending 1 A system clock source switching is pending
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FLASH_SC	FLASH State Change during mode transition indicator - This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
24 CDP_PRPH_224_255	Clock Disable Process Pending status for Peripherals 224…255 - This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
25 CDP_PRPH_ 192_223	Clock Disable Process Pending status for Peripherals 192…223 - This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
26 CDP_PRPH_ 160_191	Clock Disable Process Pending status for Peripherals 160…191 - This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
27 CDP_PRPH_ 128_159	Clock Disable Process Pending status for Peripherals 128…159 - This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
28 CDP_PRPH_96_ 127	Clock Disable Process Pending status for Peripherals 96…127 - This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_ 95	Clock Disable Process Pending status for Peripherals 64…95 - This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
	0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
30 CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32…63- This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0…31 - This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. **NOTE:** Peripheral n corresponds to the ME_PCTLn register. See the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

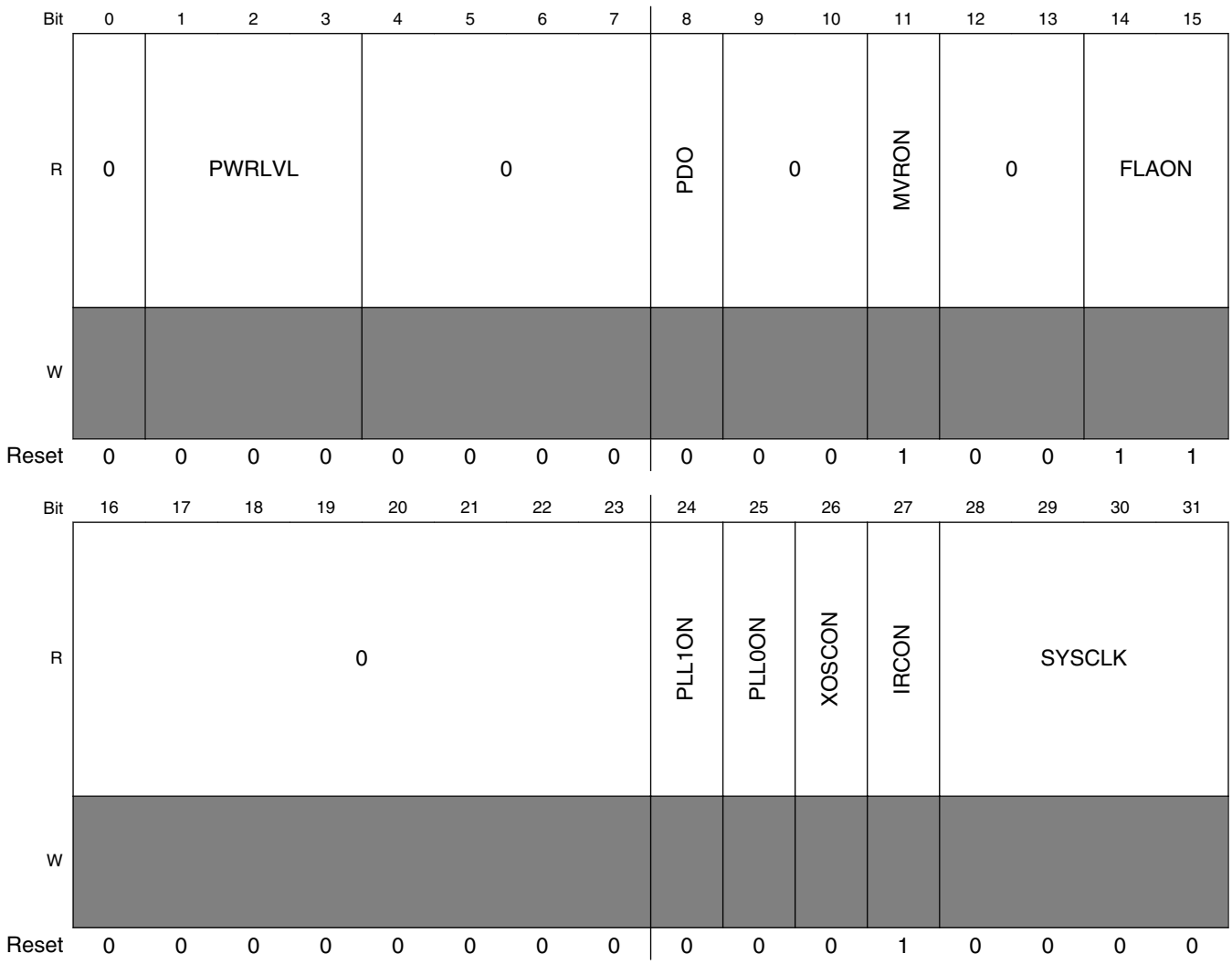
65.3.8 RESET Mode Configuration Register (MC_ME_RESET_MC)

This register configures system behavior during RESET mode.

NOTE

The following configuration values are set according to the chip configuration: XOSCON

Address: 0h base + 20h offset = 20h



MC_ME_RESET_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled **NOTE:** Power-sequence drivers control pad slopes during run mode and are disabled in certain System or User modes to further reduce power consumption. 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control **NOTE:** For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator. 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
	1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

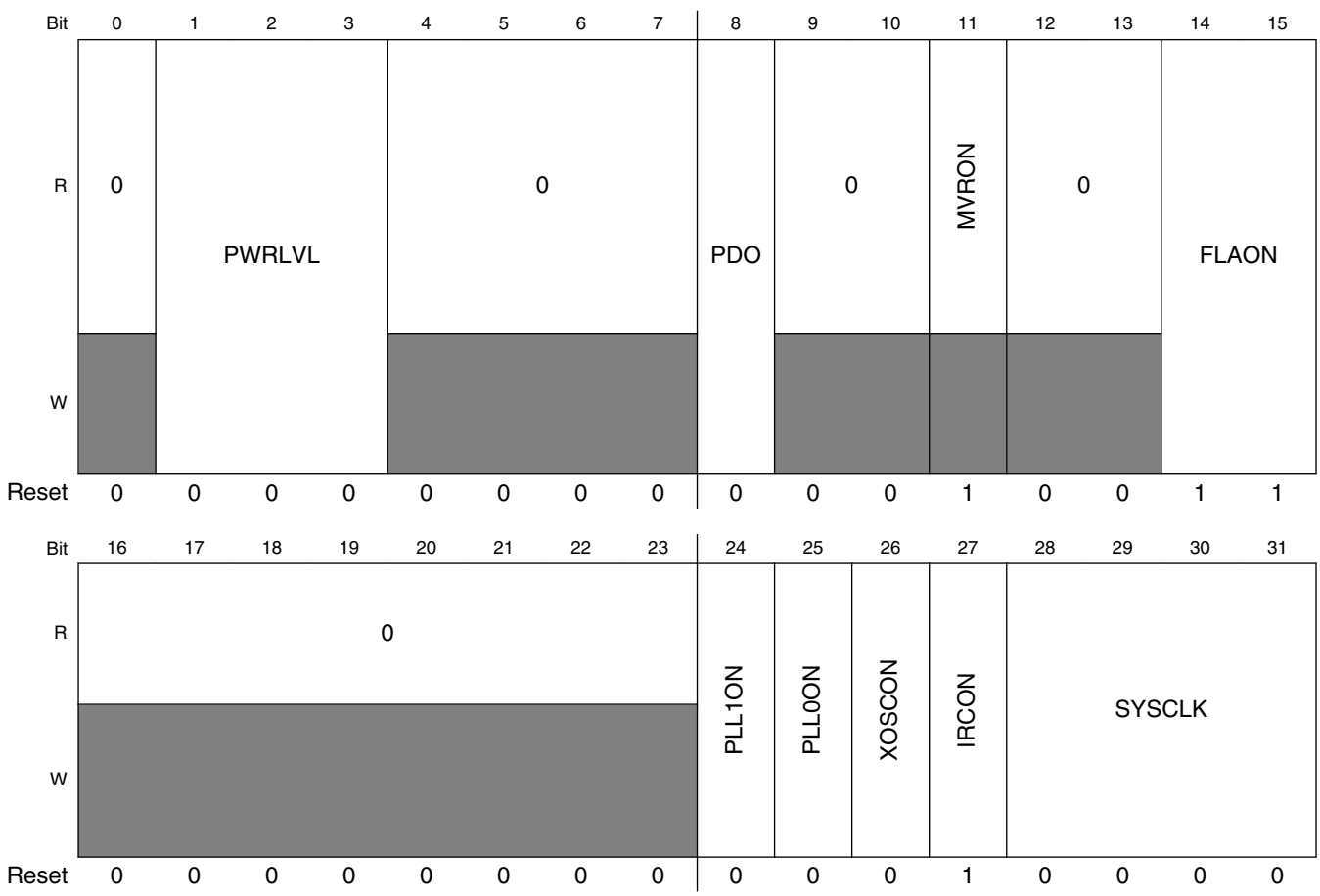
65.3.9 TEST Mode Configuration Register (MC_ME_TEST_MC)

This register configures system behavior during TEST mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 24h offset = 24h



MC_ME_TEST_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc.

Table continues on the next page...

MC_ME_TEST_MC field descriptions (continued)

Field	Description
	0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.10 SAFE Mode Configuration Register (MC_ME_SAFE_MC)

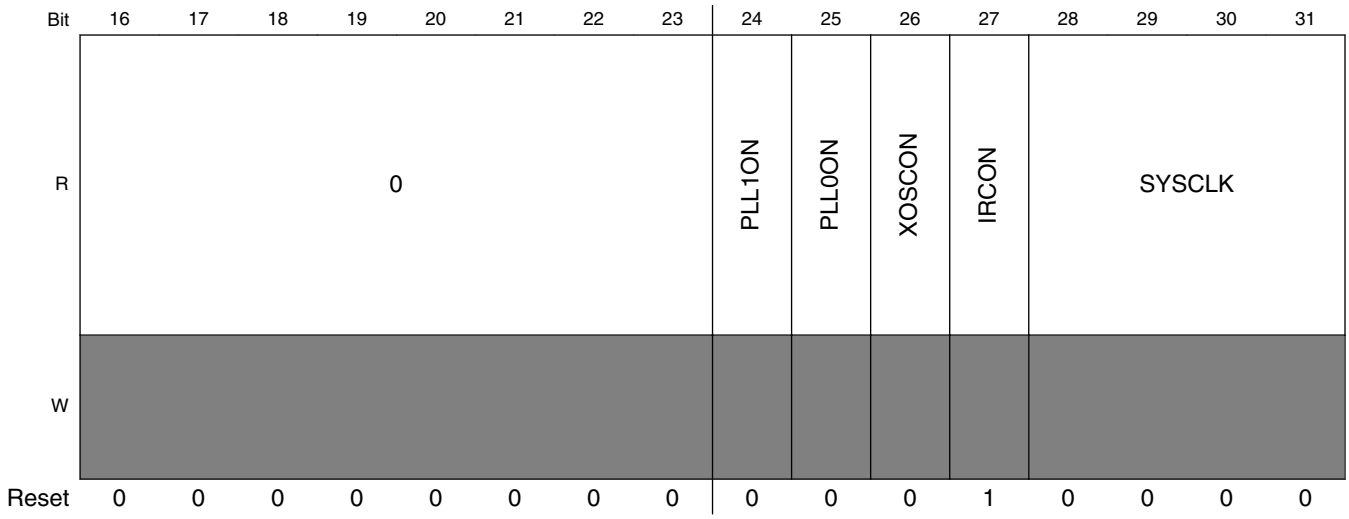
This register configures system behavior during SAFE mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1



MC_ME_SAFE_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control

Table continues on the next page...

MC_ME_SAFE_MC field descriptions (continued)

Field	Description
	0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.11 DRUN Mode Configuration Register (MC_ME_DRUN_MC)

This register configures system behavior during DRUN mode.

NOTE

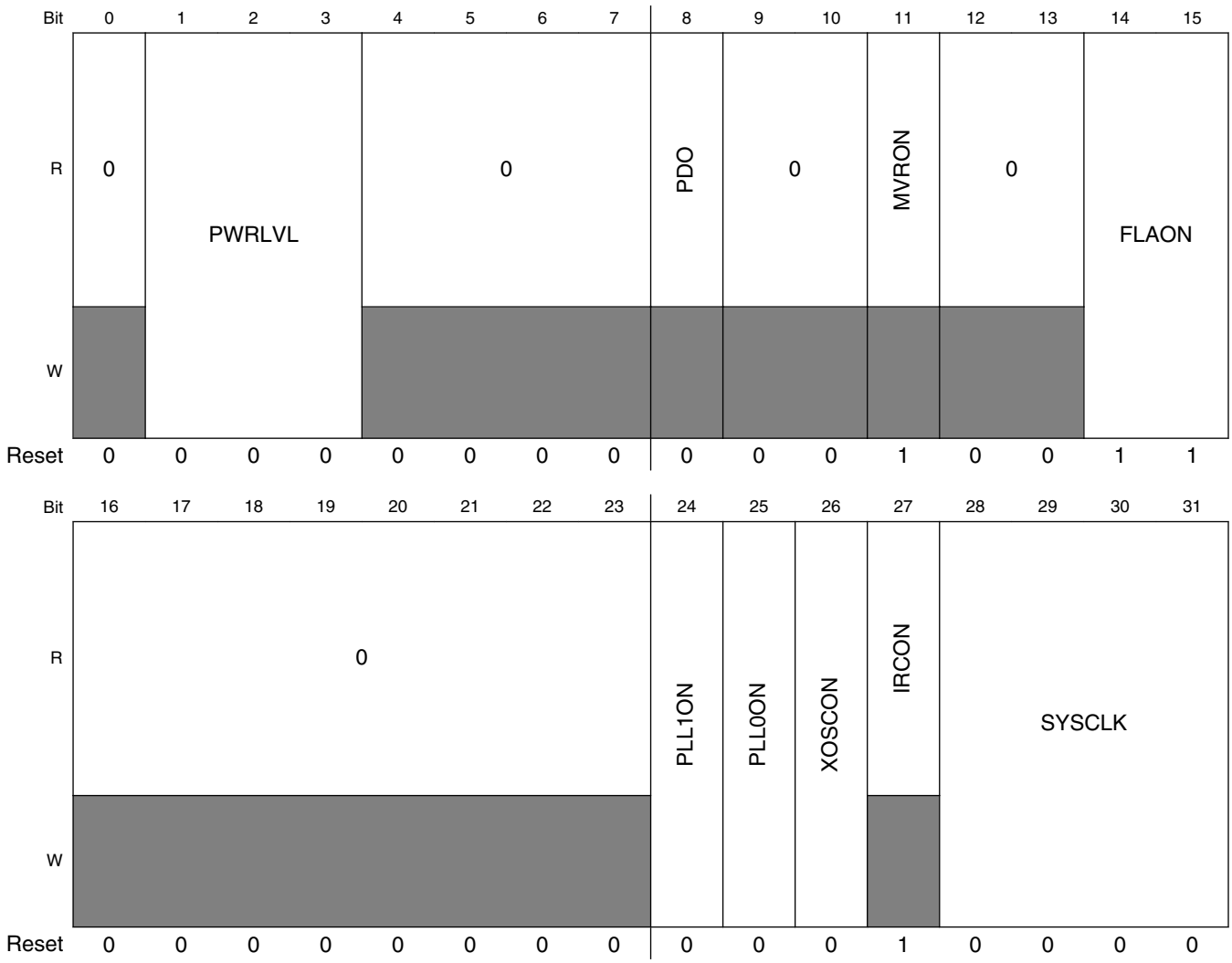
Byte write accesses are not allowed to this register.

NOTE

The following configuration values are set according to the chip configuration: XOSCON. The XOSCON bit is set according to

the DCF UTEST MISC register, XOSC_EN bit, described in the DCF chapter.

Address: 0h base + 2Ch offset = 2Ch



MC_ME_DRUN_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os.

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
	0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
	1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.12 RUN0 Mode Configuration Register (MC_ME_RUN0_MC)

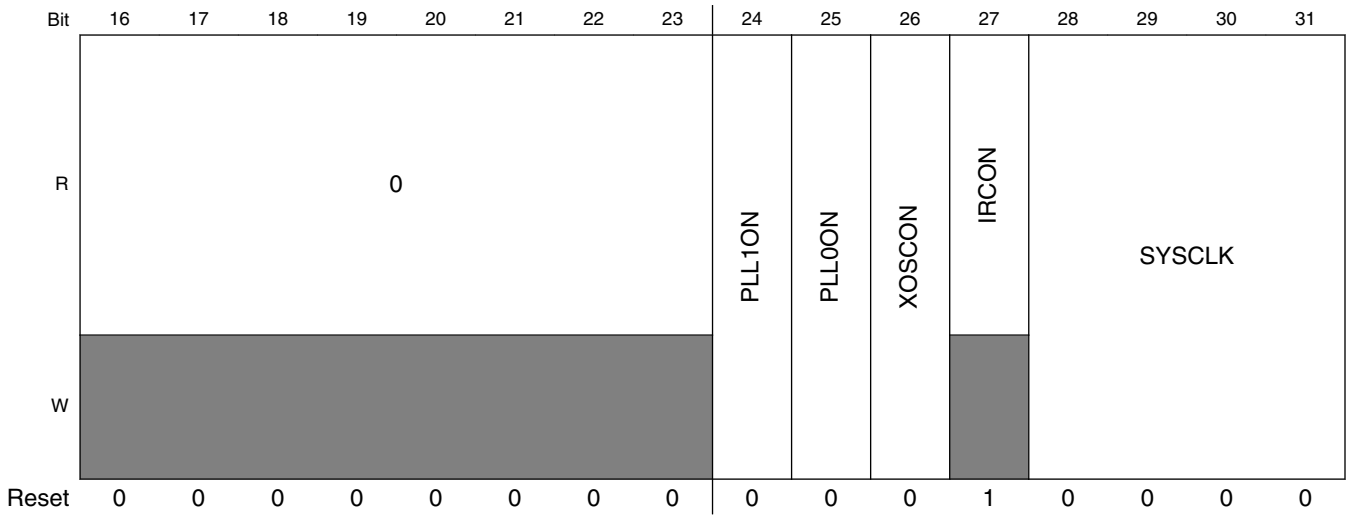
This register configures system behavior during RUN0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_RUN0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control

Table continues on the next page...

MC_ME_RUN0_MC field descriptions (continued)

Field	Description
	0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

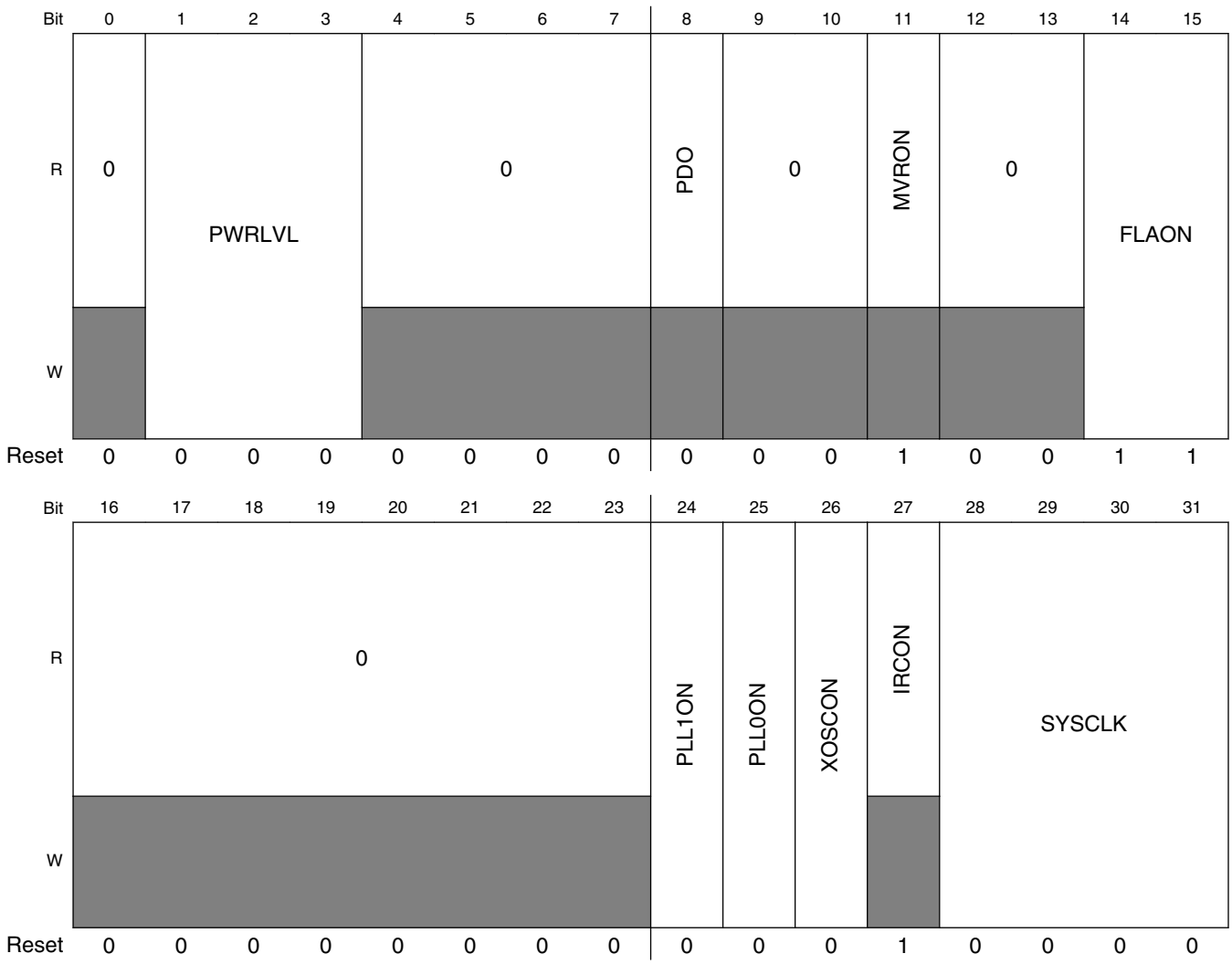
65.3.13 RUN1 Mode Configuration Register (MC_ME_RUN1_MC)

This register configures system behavior during RUN1 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 34h offset = 34h



MC_ME_RUN1_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN1_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_RUN1_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.14 RUN2 Mode Configuration Register (MC_ME_RUN2_MC)

This register configures system behavior during RUN2 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RUN2_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. Reserved
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc.

Table continues on the next page...

MC_ME_RUN2_MC field descriptions (continued)

Field	Description
	0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.15 RUN3 Mode Configuration Register (MC_ME_RUN3_MC)

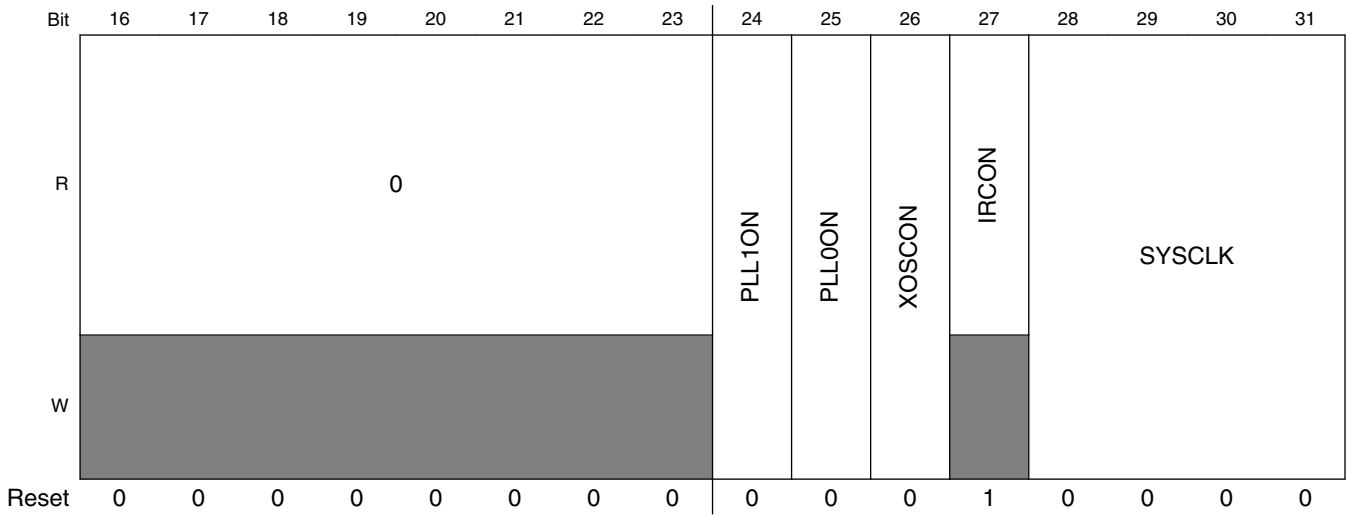
This register configures system behavior during RUN3 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_RUN3_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control

Table continues on the next page...

MC_ME_RUN3_MC field descriptions (continued)

Field	Description
	0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

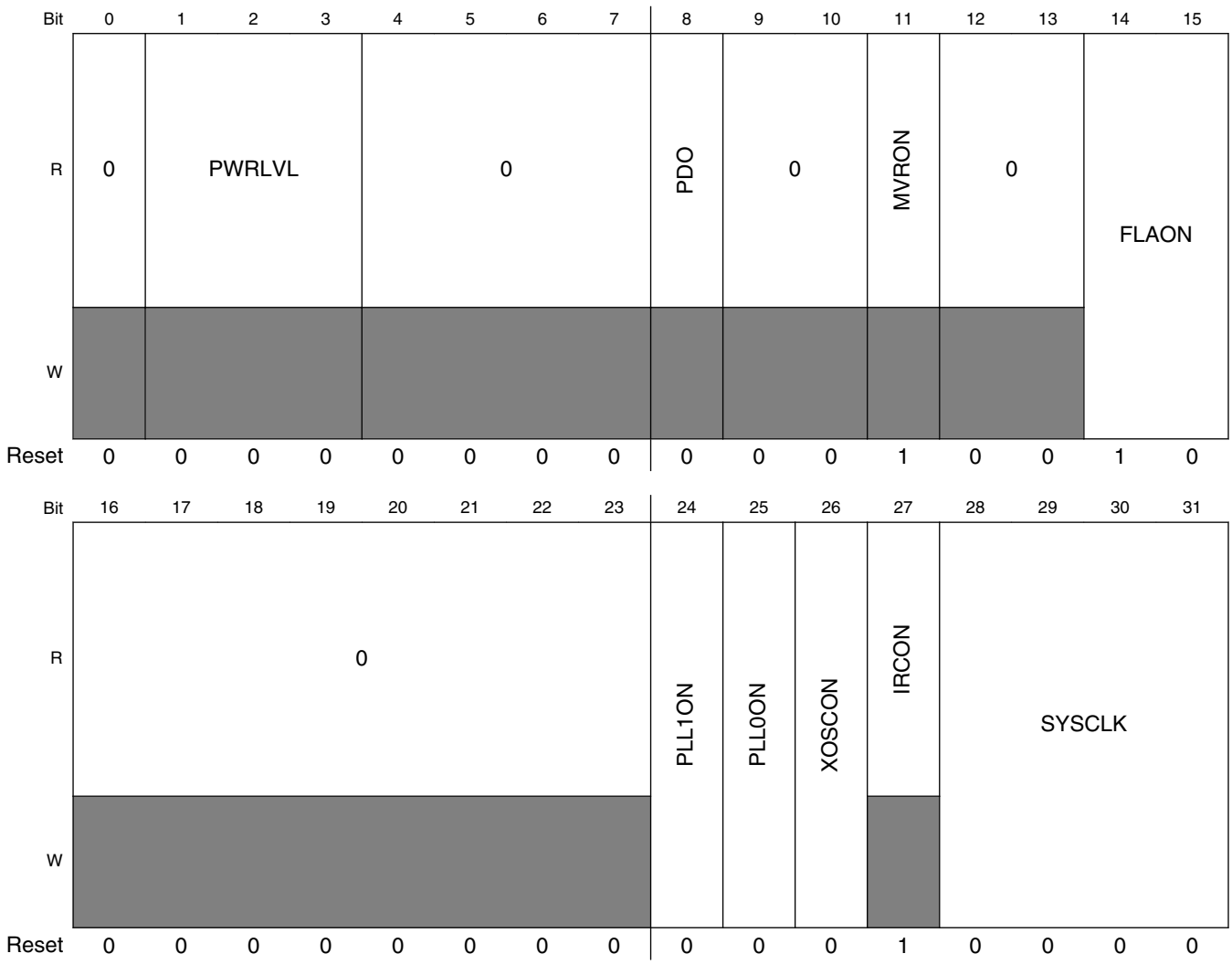
65.3.16 HALT0 Mode Configuration Register (MC_ME_HALT0_MC)

This register configures system behavior during HALT0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 40h offset = 40h



MC_ME_HALT0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_HALT0_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_HALT0_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.17 STOP0 Mode Configuration Register (MC_ME_STOP0_MC)

This register configures system behavior during STOP0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_STOP0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control[1] 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc.

Table continues on the next page...

MC_ME_STOP0_MC field descriptions (continued)

Field	Description
	0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

1. For a loss of oscillator clock, there is a dedicated monitor (CMU0) on the device. If a S/W programmable minimum frequency threshold is violated, an interrupt to the FCCU is generated and the device is safe-stated (switch to backup clock - internal RCOSC). At that point, S/W can re-enter normal mode through the MC_ME, and the oscillator is restarted. No full power down of the device is required to restart the oscillator.

65.3.18 Peripheral Status Register 0 (MC_ME_PS0)

This register provides the status of the peripherals.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_PIT_RTC_1	S_PIT_RTC_0	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SIUL	0			S_SIPI_0	0	S_LFAST_0	0					S_EBI_0	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS0 field descriptions

Field	Description
0 S_PIT_RTC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
1 S_PIT_RTC_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
2–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 S_SIUL	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 0 Peripheral is active
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_SIPI_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_LFAST_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
23–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_EBI_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.19 Peripheral Status Register 1 (MC_ME_PS1)

This register provides the status of the peripherals.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			S_ADCSD_0	S_ADCSD_2	S_ADCSD_4	S_ADCSD_6	S_ADCSD_8	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									S_CRC_0	0	S_DMAMUX_0	0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS1 field descriptions

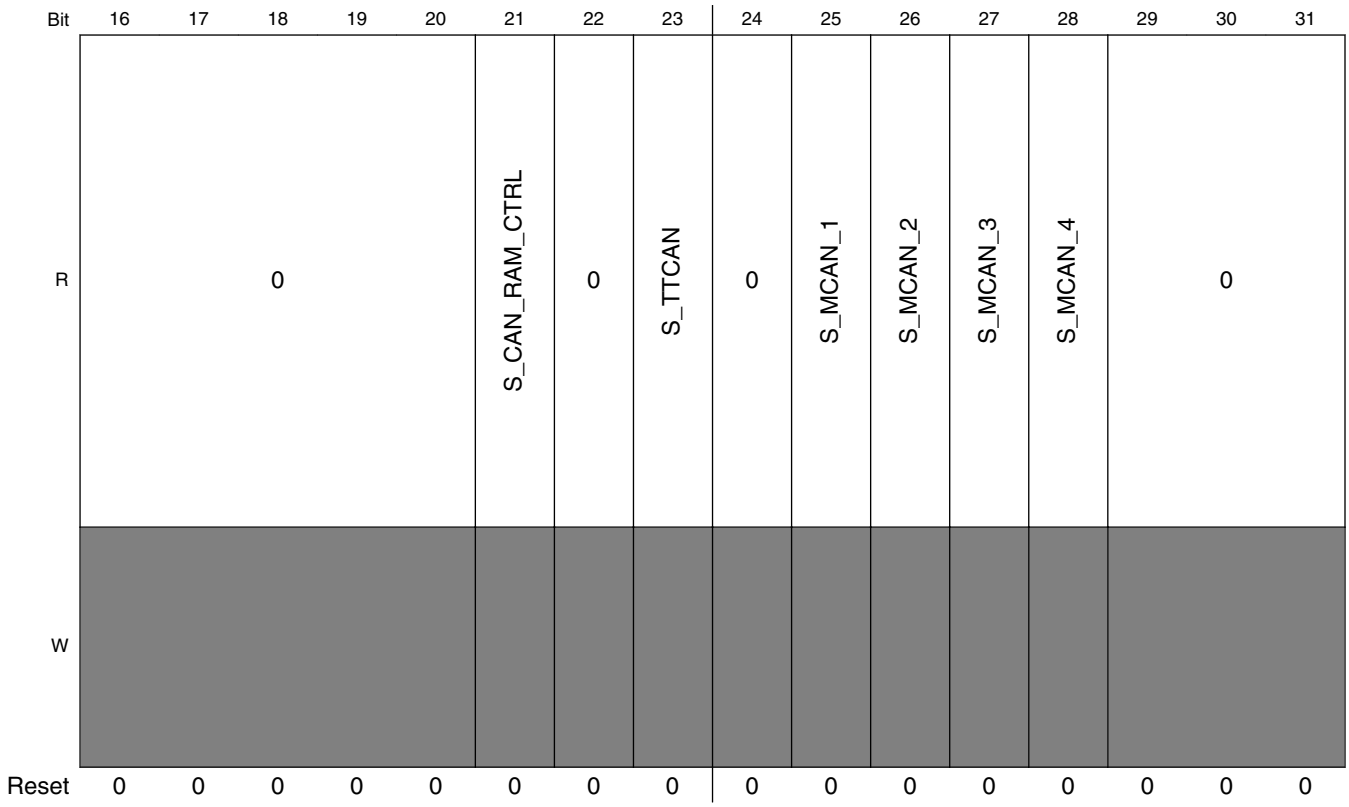
Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_ADCSD_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
4 S_ADCSD_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5 S_ADCSD_4	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
6 S_ADCSD_6	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
7 S_ADCSD_8	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
8–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_CRC_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 S_DMAMUX_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.20 Peripheral Status Register 2 (MC_ME_PS2)

This register provides the status of the peripherals.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		S_DSPI_12	S_LINFlexD_0	S_LINFlexD_1	0					S_LINFlexD_14	S_LINFlexD_16	0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MC_ME_PS2 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 S_DSPI_12	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
3 S_LINFlexD_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
4 S_LINFlexD_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 S_LINFlexD_14	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

Table continues on the next page...

MC_ME_PS2 field descriptions (continued)

Field	Description
11 S_LINFlexD_16	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
12–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 S_CAN_RAM_CTRL	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_TTCAN	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_MCAN_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
26 S_MCAN_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
27 S_MCAN_3	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
28 S_MCAN_4	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.21 Peripheral Status Register 3 (MC_ME_PS3)

This register provides the status of the peripherals.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_ADCSAR_0	0			S_ADCSAR_4	0			0							S_ADCSAR_b
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PSI5_0	0			S_FLEXRAY_0	0		S_SENT_0	0		S_IIC_0	0	S_DSPI_0	S_DSPI_1	S_DSPI_4	S_DSPI_6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS3 field descriptions

Field	Description
0 S_ADCSAR_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 S_ADCSAR_4	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 S_ADCSAR_b	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
16 S_PSI5_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_FLEXRAY_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_SENT_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_IIC_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_DSPI_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME.

Table continues on the next page...

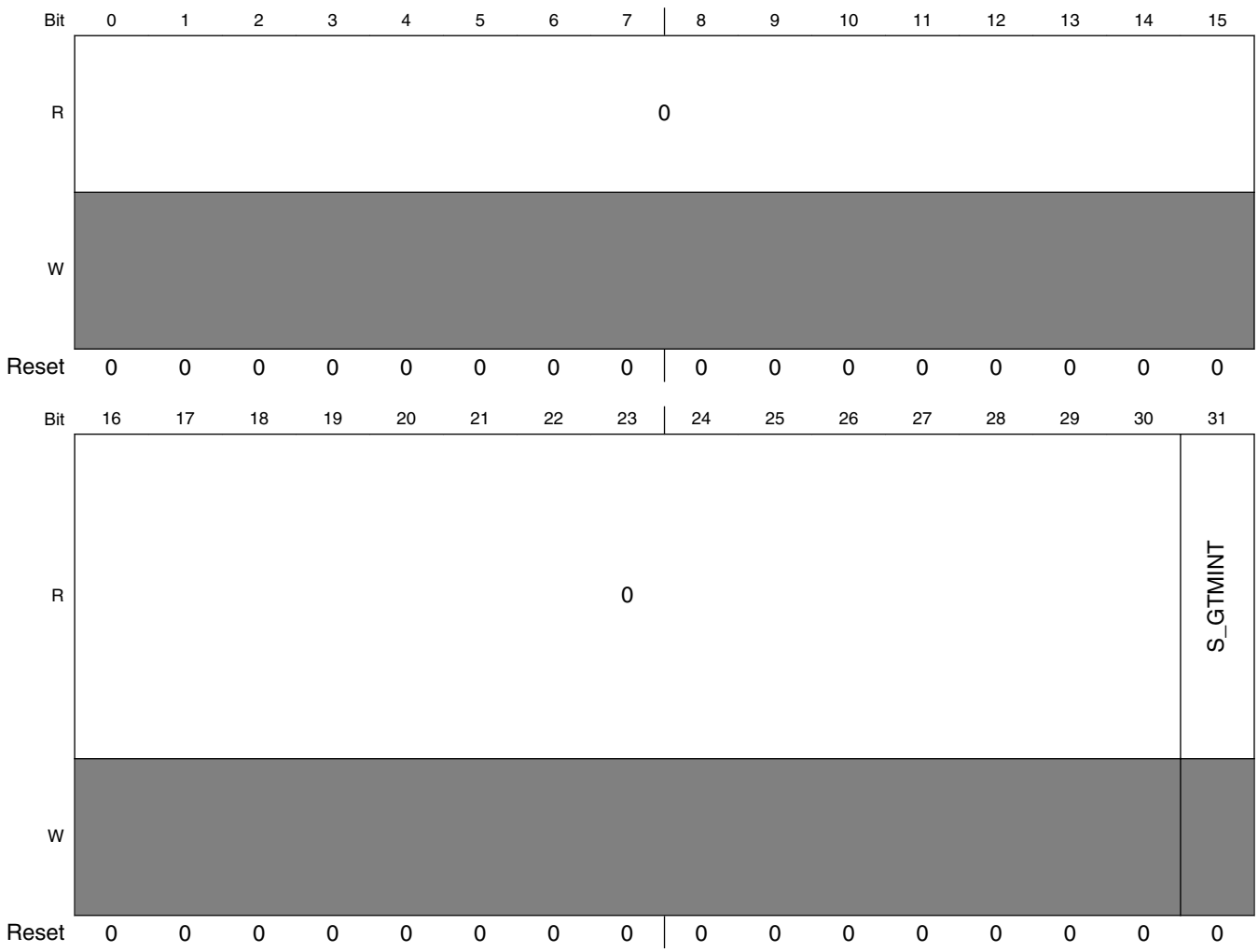
MC_ME_PS3 field descriptions (continued)

Field	Description
	0 Peripheral is frozen 1 Peripheral is active
29 S_DSPI_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
30 S_DSPI_4	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
31 S_DSPI_6	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

65.3.22 Peripheral Status Register 4 (MC_ME_PS4)

This register provides the status of the peripherals.

Address: 0h base + 70h offset = 70h



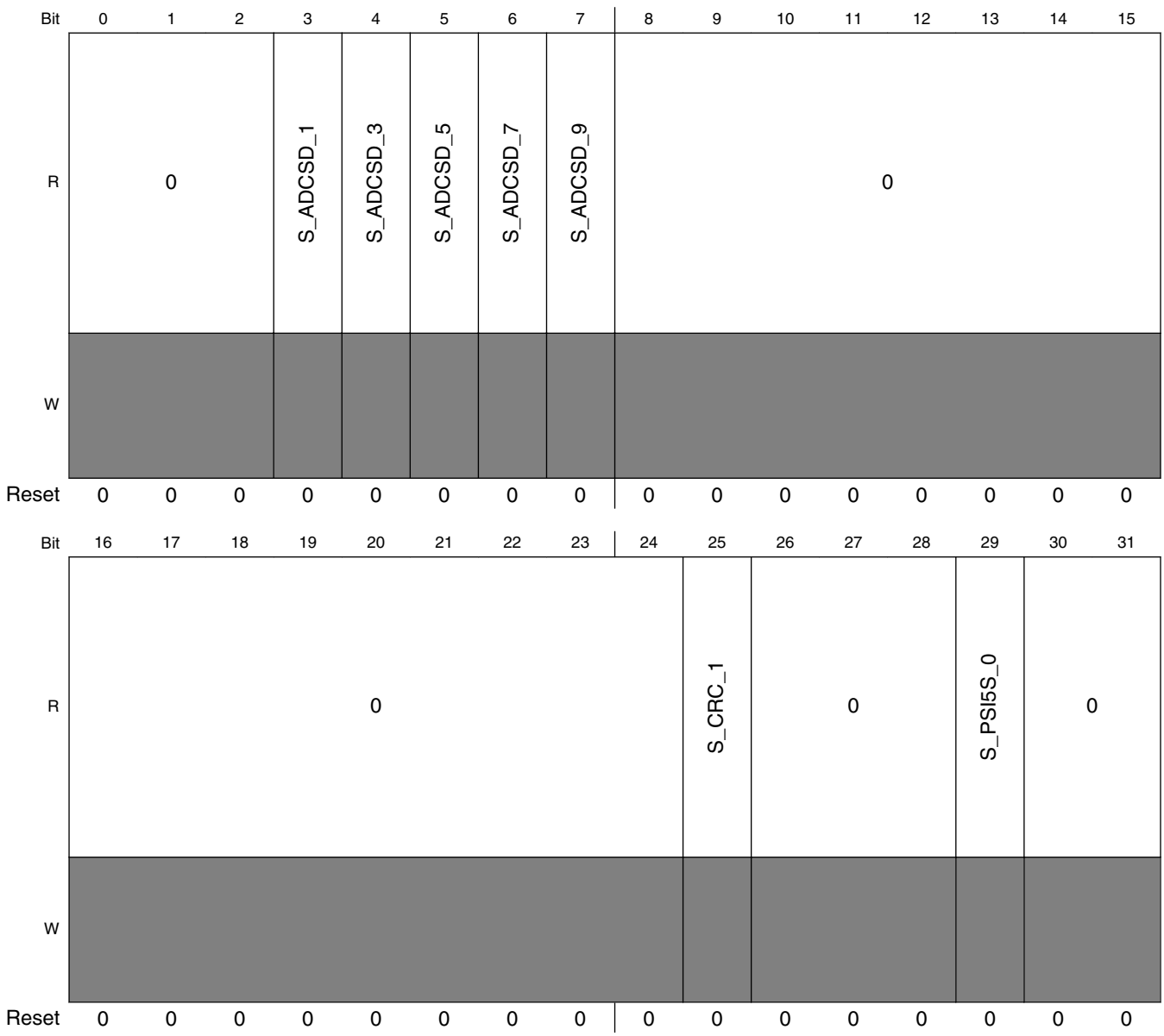
MC_ME_PS4 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 S_GTMINT	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

65.3.23 Peripheral Status Register 5 (MC_ME_PS5)

This register provides the status of the peripherals.

Address: 0h base + 74h offset = 74h



MC_ME_PS5 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

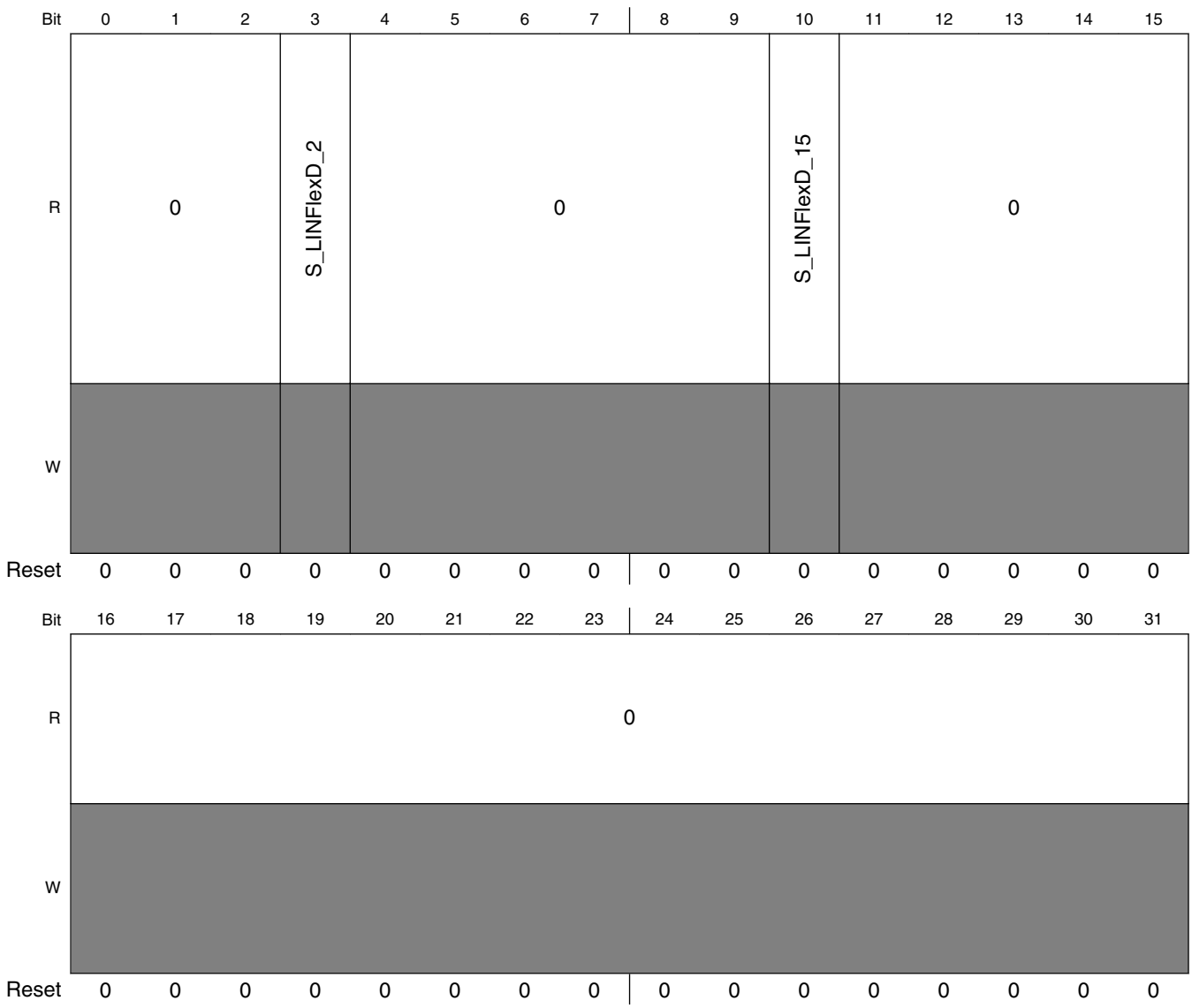
MC_ME_PS5 field descriptions (continued)

Field	Description
3 S_ADCSD_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
4 S_ADCSD_3	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5 S_ADCSD_5	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
6 S_ADCSD_7	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
7 S_ADCSD_9	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
8–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_CRC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
26–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 S_PSI5S_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.24 Peripheral Status Register 6 (MC_ME_PS6)

This register provides the status of the peripherals.

Address: 0h base + 78h offset = 78h



MC_ME_PS6 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_LINFlexD_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME.

Table continues on the next page...

MC_ME_PS6 field descriptions (continued)

Field	Description
	0 Peripheral is frozen 1 Peripheral is active
4–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 S_LINFlexD_15	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
11–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.25 Peripheral Status Register 7 (MC_ME_PS7)

This register provides the status of the peripherals.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_ADCSAR_1	S_ADCSAR_2	S_ADCSAR_3	0	S_ADCSAR_5	S_ADCSAR_6	S_ADCSAR_7	S_ADCSAR_8	S_ADCSAR_9	S_ADCSAR_10	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PSI5_1	0			S_FLEXRAY_1	0		S_SENT_1	0		S_IIC_1	0	S_DSPI_2	S_DSPI_3	S_DSPI_5	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS7 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 S_ADCSAR_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
2 S_ADCSAR_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
3 S_ADCSAR_3	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 S_ADCSAR_5	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
6 S_ADCSAR_6	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
7 S_ADCSAR_7	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
8 S_ADCSAR_8	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
9 S_ADCSAR_9	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
10 S_ADCSAR_10	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

Table continues on the next page...

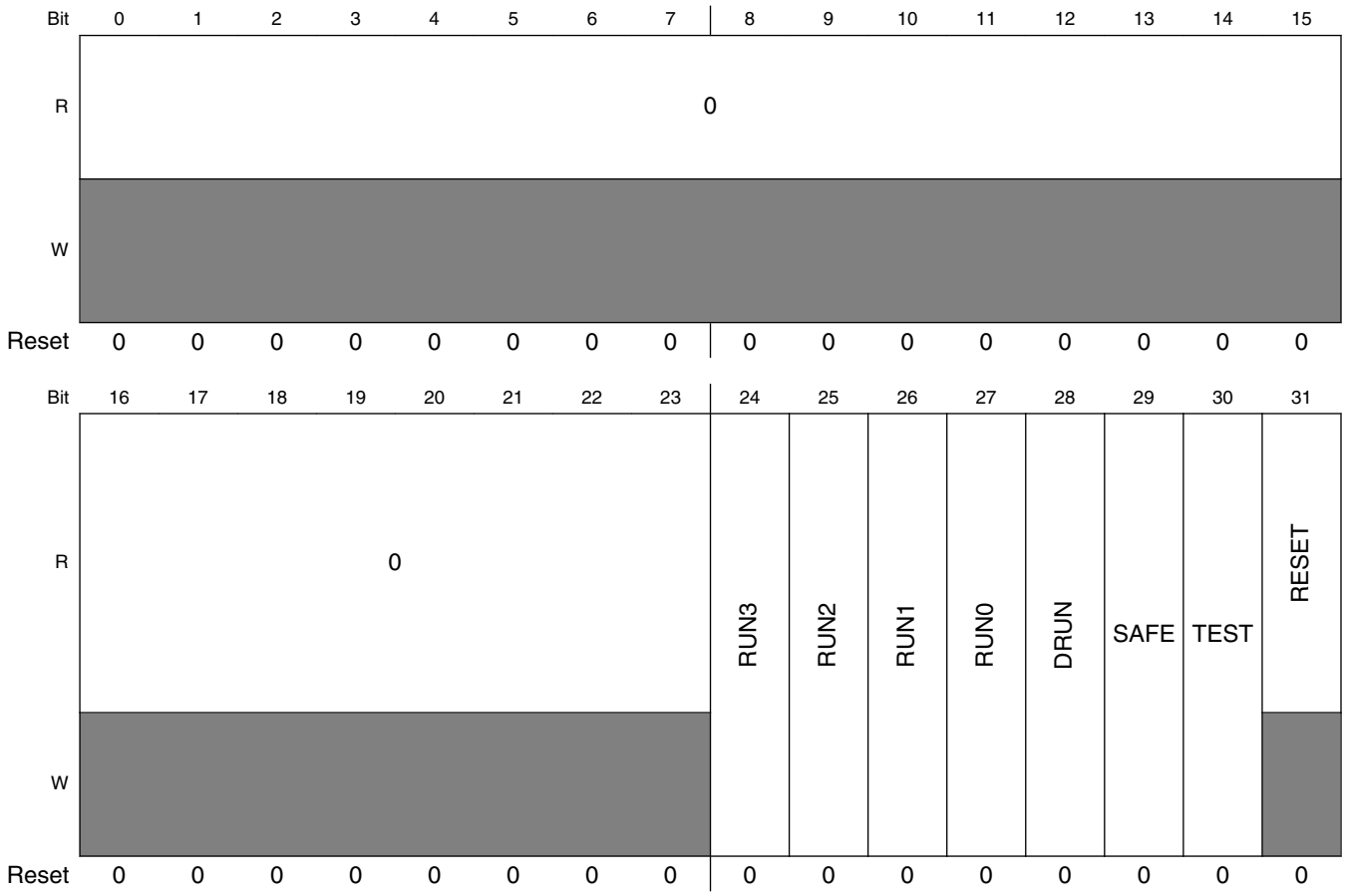
MC_ME_PS7 field descriptions (continued)

Field	Description
11–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 S_PSI5_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_FLEXRAY_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_SENT_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_IIC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_DSPI_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
29 S_DSPI_3	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
30 S_DSPI_5	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.26 Run Peripheral Configuration Register (MC_ME_RUN_PC*n*)

These registers configure eight different types of peripheral behavior during run modes.

Address: 0h base + 80h offset + (4d × i), where i=0d to 7d



MC_ME_RUN_PC*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
26 RUN1	Peripheral control during RUN1

Table continues on the next page...

MC_ME_RUN_PC*n* field descriptions (continued)

Field	Description
	0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

65.3.27 Low-Power Peripheral Configuration Register (MC_ME_LP_PC*n*)

These registers configure eight different types of peripheral behavior during non-run modes.

Address: 0h base + A0h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					STOP0	0	HALT0	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_LP_PC*n* field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

65.3.28 EBI_0 Peripheral Control Register (MC_ME_PCTL3)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + C3h offset = C3h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL3 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 1
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration

Table continues on the next page...

MC_ME_PCTL3 field descriptions (continued)

Field	Description
	100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.29 LFAST_0 Peripheral Control Register (MC_ME_PCTL9)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + C9h offset = C9h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL9 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration

Table continues on the next page...

MC_ME_PCTL9 field descriptions (continued)

Field	Description
	110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.30 SIPI_0 Peripheral Control Register (MC_ME_PCTL11)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CBh offset = CBh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL11 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration

Table continues on the next page...

MC_ME_PCTL11 field descriptions (continued)

Field	Description
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.31 SIUL Peripheral Control Register (MC_ME_PCTL15)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CFh offset = CFh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL15 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral.

Table continues on the next page...

MC_ME_PCTL15 field descriptions (continued)

Field	Description
	000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.32 PIT_RTC_0 Peripheral Control Register (MC_ME_PCTL30)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + DEh offset = DEh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL30 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration

Table continues on the next page...

MC_ME_PCTL30 field descriptions (continued)

Field	Description
	010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.33 PIT_RTC_1 Peripheral Control Register (MC_ME_PCTL31)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + DFh offset = DFh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL31 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration

Table continues on the next page...

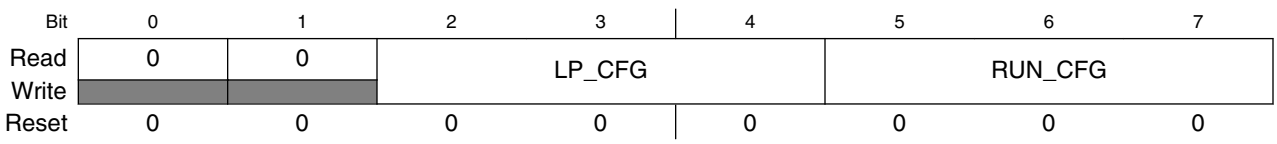
MC_ME_PCTL31 field descriptions (continued)

Field	Description
	100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.34 DMAMUX_0 Peripheral Control Register (MC_ME_PCTL36)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E4h offset = E4h



MC_ME_PCTL36 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration

Table continues on the next page...

MC_ME_PCTL36 field descriptions (continued)

Field	Description
	110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.35 CRC_0 Peripheral Control Register (MC_ME_PCTL38)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL38 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.36 ADCSD_8 Peripheral Control Register (MC_ME_PCTL56)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL56 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.37 ADCSD_6 Peripheral Control Register (MC_ME_PCTL57)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F9h offset = F9h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL57 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.38 ADCSD_4 Peripheral Control Register (MC_ME_PCTL58)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FAh offset = FAh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL58 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.39 ADCSD_2 Peripheral Control Register (MC_ME_PCTL59)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FBh offset = FBh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL59 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.40 ADCSD_0 Peripheral Control Register (MC_ME_PCTL60)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

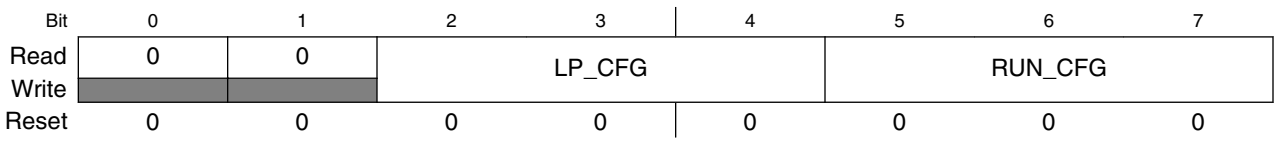
MC_ME_PCTL60 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.41 MCAN_4 Peripheral Control Register (MC_ME_PCTL67)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 103h offset = 103h



MC_ME_PCTL67 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.42 MCAN_3 Peripheral Control Register (MC_ME_PCTL68)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

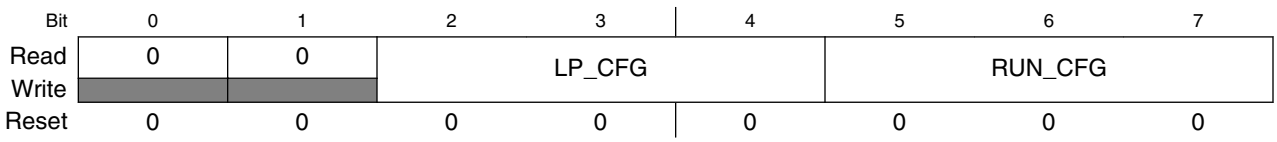
MC_ME_PCTL68 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.43 MCAN_2 Peripheral Control Register (MC_ME_PCTL69)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 105h offset = 105h



MC_ME_PCTL69 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.44 MCAN_1 Peripheral Control Register (MC_ME_PCTL70)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 106h offset = 106h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL70 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.45 TTCAN Peripheral Control Register (MC_ME_PCTL72)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL72 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.46 CAN_RAM_CTRL Peripheral Control Register (MC_ME_PCTL74)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Ah offset = 10Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL74 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.47 LINFlexD_16 Peripheral Control Register (MC_ME_PCTL84)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL84 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.48 LINFlexD_14 Peripheral Control Register (MC_ME_PCTL85)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 115h offset = 115h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL85 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.49 LINFlexD_1 Peripheral Control Register (MC_ME_PCTL91)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 11Bh offset = 11Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL91 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.50 LINFlexD_0 Peripheral Control Register (MC_ME_PCTL92)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL92 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.51 DSPI_12 Peripheral Control Register (MC_ME_PCTL93)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 11Dh offset = 11Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL93 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.52 DSPI_6 Peripheral Control Register (MC_ME_PCTL96)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL96 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.53 DSPI_4 Peripheral Control Register (MC_ME_PCTL97)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 121h offset = 121h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL97 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.54 DSPI_1 Peripheral Control Register (MC_ME_PCTL98)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 122h offset = 122h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

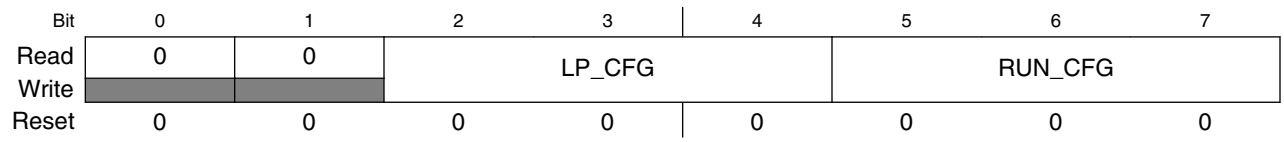
MC_ME_PCTL98 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.55 DSPI_0 Peripheral Control Register (MC_ME_PCTL99)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 123h offset = 123h



MC_ME_PCTL99 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.56 IIC_0 Peripheral Control Register (MC_ME_PCTL101)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 125h offset = 125h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL101 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.57 SENT_0 Peripheral Control Register (MC_ME_PCTL104)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 128h offset = 128h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL104 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.58 FLEXRAY_0 Peripheral Control Register (MC_ME_PCTL107)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 12Bh offset = 12Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL107 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.59 PSI5_0 Peripheral Control Register (MC_ME_PCTL111)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 12Fh offset = 12Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL111 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.60 ADCSAR_b Peripheral Control Register (MC_ME_PCTL112)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 130h offset = 130h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL112 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.61 ADCSAR_4 Peripheral Control Register (MC_ME_PCTL123)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 13Bh offset = 13Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL123 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.62 ADCSAR_0 Peripheral Control Register (MC_ME_PCTL127)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 13Fh offset = 13Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

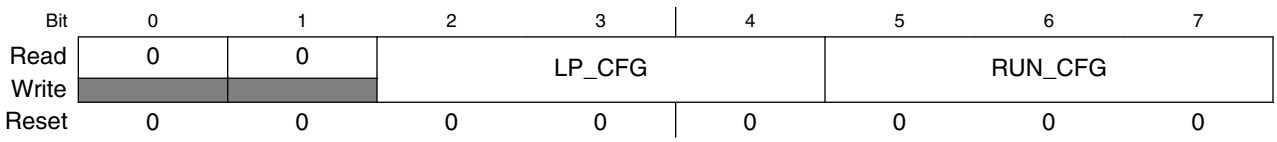
MC_ME_PCTL127 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.63 GTMINT Peripheral Control Register (MC_ME_PCTL128)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 140h offset = 140h



MC_ME_PCTL128 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.64 PSI5_S_0 Peripheral Control Register (MC_ME_PCTL162)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 162h offset = 162h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL162 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.65 CRC_1 Peripheral Control Register (MC_ME_PCTL166)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 166h offset = 166h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL166 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.66 ADCSD_9 Peripheral Control Register (MC_ME_PCTL184)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 178h offset = 178h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL184 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.67 ADCSD_7 Peripheral Control Register (MC_ME_PCTL185)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 179h offset = 179h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL185 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.68 ADCSD_5 Peripheral Control Register (MC_ME_PCTL186)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 17Ah offset = 17Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL186 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.69 ADCSD_3 Peripheral Control Register (MC_ME_PCTL187)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 17Bh offset = 17Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL187 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.70 ADCSD_1 Peripheral Control Register (MC_ME_PCTL188)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 17Ch offset = 17Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL188 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.71 LINFlexD_15 Peripheral Control Register (MC_ME_PCTL213)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 195h offset = 195h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL213 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.72 LINFlexD_2 Peripheral Control Register (MC_ME_PCTL220)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 19Ch offset = 19Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL220 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.73 DSPI_5 Peripheral Control Register (MC_ME_PCTL225)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1A1h offset = 1A1h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL225 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.74 DSPI_3 Peripheral Control Register (MC_ME_PCTL226)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1A2h offset = 1A2h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

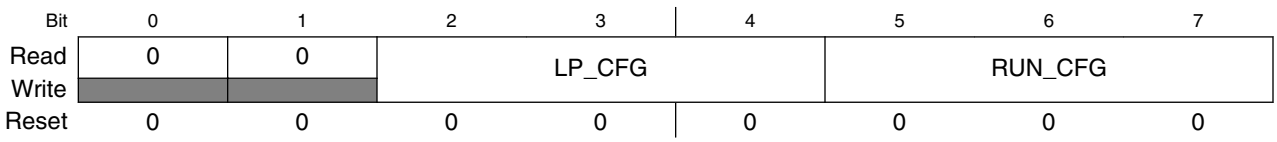
MC_ME_PCTL226 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.75 DSPI_2 Peripheral Control Register (MC_ME_PCTL227)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1A3h offset = 1A3h



MC_ME_PCTL227 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.76 IIC_1 Peripheral Control Register (MC_ME_PCTL229)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1A5h offset = 1A5h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL229 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.77 SENT_1 Peripheral Control Register (MC_ME_PCTL232)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL232 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.78 FLEXRAY_1 Peripheral Control Register (MC_ME_PCTL235)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1ABh offset = 1ABh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL235 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.79 PSI5_1 Peripheral Control Register (MC_ME_PCTL239)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1AFh offset = 1AFh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL239 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.80 ADCSAR_10 Peripheral Control Register (MC_ME_PCTL245)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1B5h offset = 1B5h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL245 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.81 ADCSAR_9 Peripheral Control Register (MC_ME_PCTL246)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1B6h offset = 1B6h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL246 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.82 ADCSAR_8 Peripheral Control Register (MC_ME_PCTL247)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1B7h offset = 1B7h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL247 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.83 ADCSAR_7 Peripheral Control Register (MC_ME_PCTL248)

This register selects the configurations during run and non-run modes for the associated peripheral.

NOTE

After modifying any of the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers, software must request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behavior between the peripheral clock control process and the clock status reporting in the ME_PSn registers.

Address: 0h base + 1B8h offset = 1B8h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL248 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.84 ADCSAR_6 Peripheral Control Register (MC_ME_PCTL249)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1B9h offset = 1B9h

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL249 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.85 ADCSAR_5 Peripheral Control Register (MC_ME_PCTL250)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1BAh offset = 1BAh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL250 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.86 ADCSAR_3 Peripheral Control Register (MC_ME_PCTL252)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1BCh offset = 1BCh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL252 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.87 ADCSAR_2 Peripheral Control Register (MC_ME_PCTL253)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1BDh offset = 1BDh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL253 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.88 ADCSAR_1 Peripheral Control Register (MC_ME_PCTL254)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1BEh offset = 1BEh

Bit	0	1	2	3	4	5	6	7
Read	0	0	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

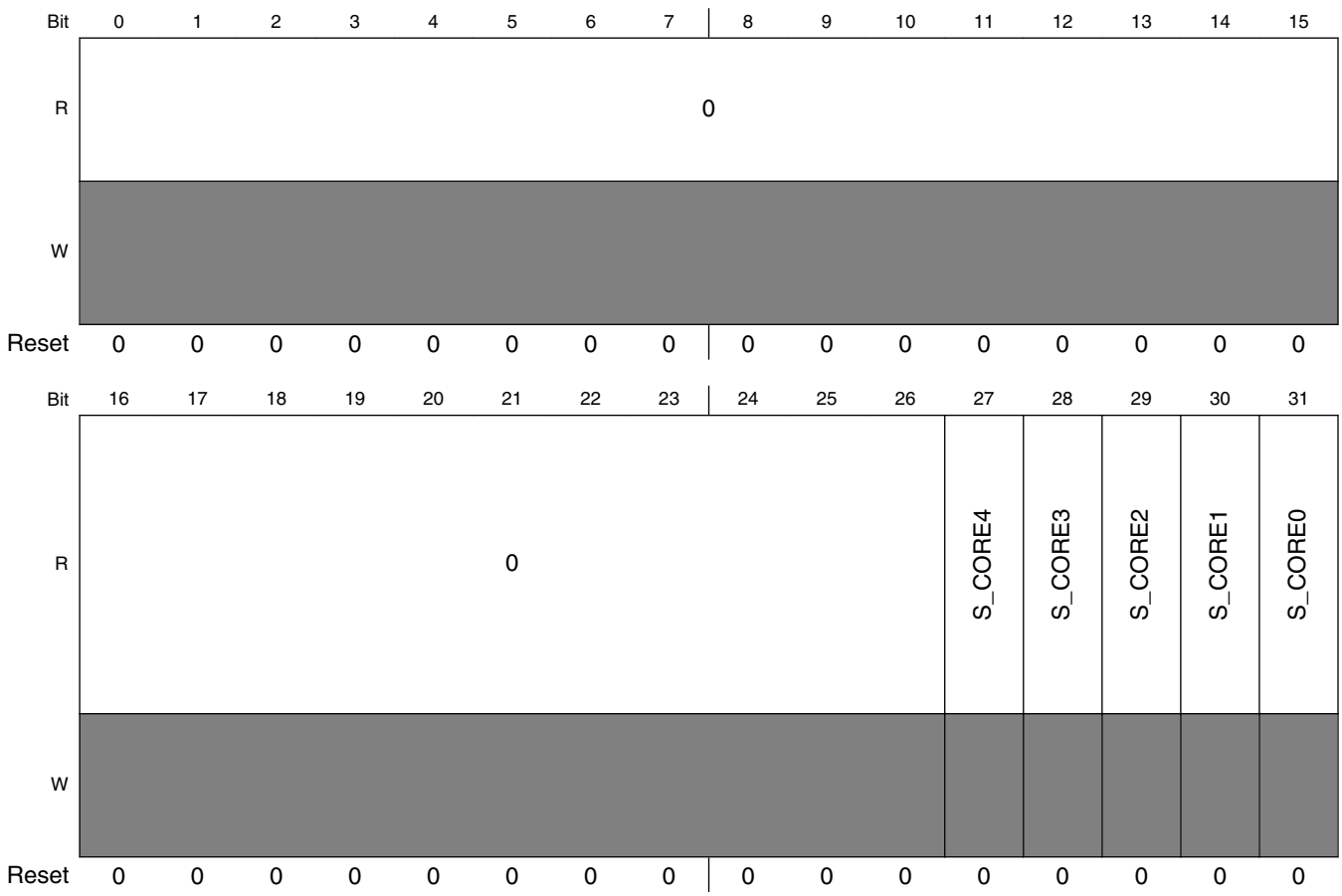
MC_ME_PCTL254 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

65.3.89 Core Status Register (MC_ME_CS)

This register provides the status of each core.

Address: 0h base + 1C0h offset = 1C0h



MC_ME_CS field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 S_CORE4	Core 4 status - This bits specifies the current status of the HSM which is controlled by the ME_CCTL4 register. 0 the HSM is disabled 1 the HSM is running
28 S_CORE3	Core 3 status - This bits specifies the current status of main core_1 which is controlled by the ME_CCTL3 register. 0 main core_1 is disabled 1 main core_1 is running

Table continues on the next page...

MC_ME_CS field descriptions (continued)

Field	Description
29 S_CORE2	Core 2 status - This bits specifies the current status of checker core_0s which is controlled by the ME_CCTL2 register. 0 checker core_0s is disabled 1 checker core_0s is running
30 S_CORE1	Core 1 status - This bits specifies the current status of main core_0 which is controlled by the ME_CCTL1 register. 0 main core_0 is disabled 1 main core_0 is running
31 S_CORE0	Core 0 status - This bits specifies the current status of peripheral core_2 which is controlled by the ME_CCTL0 register. 0 peripheral core_2 is disabled 1 peripheral core_2 is running

65.3.90 CORE0 Core Control Register (MC_ME_CCTL0)

This register controls whether peripheral core_2 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1C4h offset = 1C4h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	1	1	1	1	1	1	1	0

MC_ME_CCTL0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - peripheral core_2 is always disabled during STOP0.

Table continues on the next page...

MC_ME_CCTL0 field descriptions (continued)

Field	Description
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - peripheral core_2 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
9 RUN2	Core control during RUN2 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
10 RUN1	Core control during RUN1 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
11 RUN0	Core control during RUN0 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
12 DRUN	Core control during DRUN 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
13 SAFE	Core control during SAFE 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
14 TEST	Core control during TEST 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
15 RESET	Core control during RESET - peripheral core_2 is always disabled during RESET.

65.3.91 CORE1 Core Control Register (MC_ME_CCTL1)

This register controls whether main core_0 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

NOTE

It is possible to turn off Master Core in any RUN mode transition. It is software responsibility that at least 1 core should be on in that RUN mode and all software routines are configured, such that all requests to the master core are now routed to new assigned master core.

Address: 0h base + 1C6h offset = 1C6h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_CCTL1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - main core_0 is always disabled during STOP0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - main core_0 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 main core_0 is disabled with clock gated 1 main core_0 is running
9 RUN2	Core control during RUN2 0 main core_0 is disabled with clock gated 1 main core_0 is running
10 RUN1	Core control during RUN1 0 main core_0 is disabled with clock gated 1 main core_0 is running
11 RUN0	Core control during RUN0 0 main core_0 is frozen with clock gated 1 main core_0 is running
12 DRUN	Core control during DRUN

Table continues on the next page...

MC_ME_CCTL1 field descriptions (continued)

Field	Description
	0 main core_0 is frozen with clock gated 1 main core_0 is running
13 SAFE	Core control during SAFE 0 main core_0 is frozen with clock gated 1 main core_0 is running
14 TEST	Core control during TEST 0 main core_0 is frozen with clock gated 1 main core_0 is running
15 RESET	Core control during RESET - main core_0 is always disabled during RESET.

65.3.92 CORE2 Core Control Register (MC_ME_CCTL2)

This register controls whether checker core_0s is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1C8h offset = 1C8h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_CCTL2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - checker core_0s is always disabled during STOP0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_CCTL2 field descriptions (continued)

Field	Description
7 HALT0	Core control during HALT0 - checker core_0s is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 checker core_0s is disabled with clock gated 1 checker core_0s is running
9 RUN2	Core control during RUN2 0 checker core_0s is disabled with clock gated 1 checker core_0s is running
10 RUN1	Core control during RUN1 0 checker core_0s is disabled with clock gated 1 checker core_0s is running
11 RUN0	Core control during RUN0 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
12 DRUN	Core control during DRUN 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
13 SAFE	Core control during SAFE 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
14 TEST	Core control during TEST 0 checker core_0s is frozen with clock gated 1 checker core_0s is running
15 RESET	Core control during RESET - checker core_0s is always disabled during RESET.

65.3.93 CORE3 Core Control Register (MC_ME_CCTL3)

This register controls whether main core_1 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1CAh offset = 1CAh

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_CCTL3 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - main core_1 is always disabled during STOP0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - main core_1 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 main core_1 is disabled with clock gated 1 main core_1 is running
9 RUN2	Core control during RUN2 0 main core_1 is disabled with clock gated 1 main core_1 is running
10 RUN1	Core control during RUN1 0 main core_1 is disabled with clock gated 1 main core_1 is running
11 RUN0	Core control during RUN0 0 main core_1 is frozen with clock gated 1 main core_1 is running
12 DRUN	Core control during DRUN 0 main core_1 is frozen with clock gated 1 main core_1 is running
13 SAFE	Core control during SAFE 0 main core_1 is frozen with clock gated 1 main core_1 is running
14 TEST	Core control during TEST 0 main core_1 is frozen with clock gated 1 main core_1 is running

Table continues on the next page...

MC_ME_CCTL3 field descriptions (continued)

Field	Description
15 RESET	Core control during RESET - main core_1 is always disabled during RESET.

65.3.94 CORE4 Core Control Register (MC_ME_CCTL4)

This register controls whether the HSM is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1CCh offset = 1CCh

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_CCTL4 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - the HSM is always disabled during STOP0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - the HSM is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 the HSM is disabled with clock gated 1 the HSM is running
9 RUN2	Core control during RUN2 0 the HSM is disabled with clock gated 1 the HSM is running

Table continues on the next page...

MC_ME_CCTL4 field descriptions (continued)

Field	Description
10 RUN1	Core control during RUN1 0 the HSM is disabled with clock gated 1 the HSM is running
11 RUN0	Core control during RUN0 0 the HSM is frozen with clock gated 1 the HSM is running
12 DRUN	Core control during DRUN 0 the HSM is frozen with clock gated 1 the HSM is running
13 SAFE	Core control during SAFE 0 the HSM is frozen with clock gated 1 the HSM is running
14 TEST	Core control during TEST 0 the HSM is frozen with clock gated 1 the HSM is running
15 RESET	Core control during RESET - the HSM is always disabled during RESET.

65.3.95 CORE0 Core Address Register (MC_ME_CADDR0)

This registers gives the boot address for peripheral core_2 and a bit for controlling whether peripheral core_2 is to be reset on the next mode change that has peripheral core_2 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1').

NOTE

The reset value of the ADDR field of the ME_CADDR0 is determined by the chip configuration and boot mode.

Address: 0h base + 1E0h offset = 1E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR														0	RMC
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	0	0

* Notes:
- x = Undefined at reset.

MC_ME_CADDR0 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by peripheral core_2 as the boot address (32-bit word aligned) when peripheral core_2 next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode. 0 peripheral core_2 will not be reset on the next mode change 1 peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode

65.3.96 CORE1 Core Address Register (MC_ME_CADDR1)

This registers gives the boot address for main core_0 and a bit for controlling whether main core_0 is to be reset on the next mode change that has main core_0 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1').

Address: 0h base + 1E4h offset = 1E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR														0	RMC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_CADDR1 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by main core_0 as the boot address (32-bit word aligned) when main core_0 next exits reset.

Table continues on the next page...

MC_ME_CADDR1 field descriptions (continued)

Field	Description
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode. 0 main core_0 will not be reset on the next mode change 1 main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode

65.3.97 CORE2 Core Address Register (MC_ME_CADDR2)

This registers contains a bit for controlling whether checker core_0s is to be reset on the next mode change that has checker core_0s configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1').

Address: 0h base + 1E8h offset = 1E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															RMC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_CADDR2 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change — checker core_0s will be reset on the next mode change that has checker core_0s configured to be running in the target mode. 0 checker core_0s will not be reset on the next mode change 1 checker core_0s will be reset on the next mode change that has checker core_0s configured to be running in the target mode

65.3.98 CORE3 Core Address Register (MC_ME_CADDR3)

This registers gives the boot address for main core_1 and a bit for controlling whether main core_1 is to be reset on the next mode change that has main core_1 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1').

Address: 0h base + 1ECh offset = 1ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_CADDR3 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by main core_1 as the boot address (32-bit word aligned) when main core_1 next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - main core_1 will be reset on the next mode change that has main core_1 configured to be running in the target mode. 0 main core_1 will not be reset on the next mode change 1 main core_1 will be reset on the next mode change that has main core_1 configured to be running in the target mode

65.3.99 CORE4 Core Address Register (MC_ME_CADDR4)

This registers gives the boot address for the HSM and a bit for controlling whether the HSM is to be reset on the next mode change that has the HSM configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1').

NOTE

The reset value of the ADDR field of the ME_CADDR4 is determined by the chip configuration.

NOTE

Before setting the RMC bit and issuing a Mode Change to reset the HSM Core, make sure that all the HSM-HOST Bridge registers, on the HOST side are moved to the RESET state.

Address: 0h base + 1F0h offset = 1F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR														0	RMC
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	0	0

* Notes:
- x = Undefined at reset.

MC_ME_CADDR4 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by the HSM as the boot address (32-bit word aligned) when main core_0 next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - The HSM will be reset on the next mode change that has the HSM configured to be running in the target mode. 0 the HSM will not be reset on the next mode change 1 the HSM will be reset on the next mode change that has the HSM configured to be running in the target mode

65.4 Functional description

65.4.1 Mode transition request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register.The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Mode transition interrupts.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as RUN0…3 → RUN0…3, DRUN → DRUN, SAFE → SAFE, and TEST → TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set until the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note

It is recommended that software polls the S_MTRANS bit in the ME_GS register after requesting a transition to HALT0 or STOP0 modes.

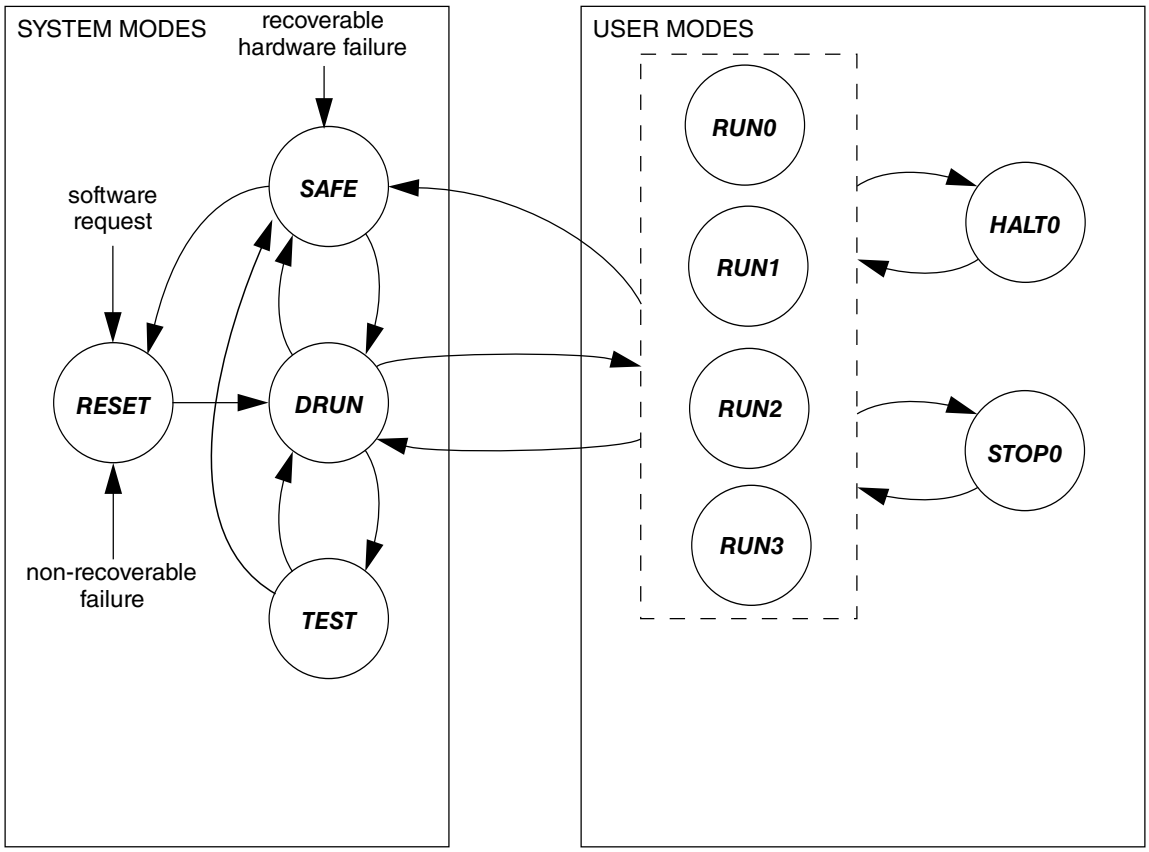


Figure 65-2. MC_ME Mode Diagram

65.4.2 Modes details

65.4.2.1 RESET Mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0…3, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with either "0000" for a 'functional' reset or "1111" for a 'destructive' reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

65.4.2.2 DRUN Mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0…3, SAFE, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0011"

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

65.4.2.3 SAFE Mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with "0010"
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the Reset Generation Module (MC_RGM) chapter for details)

Note

If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note

If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the DRUN mode, or
  - completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. The input levels remain unchanged.

65.4.2.4 TEST Mode

The chip enters this mode on the following events:

- from the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0001"

As soon as any of the above events has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to "1111", and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

65.4.2.5 RUN0…3 Modes

The chip enters one of these modes on the following events:

- from the DRUN, SAFE, or another RUN0…3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0100…0111"
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0…3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0…3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

65.4.2.6 HALT0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1000".

As soon as any of the above events has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. The flash can be put in low-power or power-down mode as needed. If there is a HALT0 mode request while an interrupt request is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen the additional core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

Note

It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on HALT0 mode exit to ensure that the previous RUN0…3 mode configuration has been fully restored before executing critical code.

65.4.2.7 STOP0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1010".

As soon as any of the above events has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the primary PLL
- the secondary PLL

The flash can be put in power-down mode as needed. If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the ***STOP0*** mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note

It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0…3 mode configuration has been fully restored before executing critical code.

65.4.3 Mode transition process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

65.4.3.1 Target mode request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Mode transition interrupts for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

65.4.3.2 Target mode configuration loading

On completion of the Target mode request step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in . A '√' indicates that a given resource is configurable for a given mode.

Table 65-2. MC_ME Resource Control Overview (continued)

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0…3	HALT0	STOP0
IRC	on	√ on	on	on	on	on	on
XOSC	off	√ off	off	√ off	√ off	√ off	√ off
PLL0	off	√ off	off	√ off	√ off	√ off	off
PLL1	off	√ off	off	√ off	√ off	√ off	√ off
FLASH	normal	√ normal	normal	√ normal	√ normal	√ low-power	√ power-down
MVREG	on	on	on	on	on	on	on
PDO	off	√ off	√ on	off	off	off	√ off

65.4.3.3 Peripheral clocks disable

On completion of the Target mode request step and the system clock frequency ramp-down of the System clock switching step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0…7 and ME_LP_PC0…7, and the peripheral control registers ME_PCTLn

Note

The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Peripheral clock gating for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

65.4.3.4 Processor low-power mode entry

If, on completion of the Peripheral clocks disable step, the mode transition is to the HALT0,STOP0 mode, the MC_ME requests each processor to enter its stopped state. Each processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the Peripheral clocks disable step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests each processor which is configured in the ME_CCTLn registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

65.4.3.5 Processor Clocks Disable

If, on completion of the Processor low-power mode entry step, and all processors which are configured in the ME_CCTLn registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

warning

Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

65.4.3.6 System clock disable

If, on completion of the Processor Clocks Disable step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

65.4.3.7 Clock sources switch-on

On completion of the Processor low-power mode entry step, the MC_ME switches on all clock sources based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

65.4.3.8 Flash module switch-on

On completion of the step, if the flash needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flash is available for access, the S_FLA bit field of the ME_GS register is updated to "11" by hardware.

warning

It is illegal to switch the FLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

65.4.3.9 Pad outputs-on

On completion of the step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- the slew rate control mechanism is switched on

65.4.3.10 Peripheral clocks enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0…7, ME_LP_PC0…7, and the peripheral control registers ME_PCTLn, the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed, and if the value of the PWRLVL field of the ME_<target mode>_MC register is process is completed.

65.4.3.11 System and processor clocks enable

On completion of the Flash module switch-on step, the MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the ME_CCTLn registers to be running in the target mode and which have their RMC bits in the ME_CADDRn registers set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the S_MTRANS bit of the MW_GS register at which time they are deasserted.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register's, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the System clock switching process is completed.

65.4.3.12 Processor low-power mode exit

On completion of the System and processor clocks enable step, the MC_ME requests all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode.

65.4.3.13 System clock switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the external crystal osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the primary PLL (PHI) takes effect only after the S_XOSC bit of the ME_GS register is set by hardware (i.e., the external crystal oscillator has stabilized).
- The target clock configuration for the secondary PLL takes effect only after the S_PLL0 bit of the ME_GS register is set by hardware (i.e., the primary PLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with "1111". This is possible only in the TEST mode.

If the value of the PWRLVL field of the ME_<target mode>_MC register not equal to that of the ME_<current mode>_MC register's, a progressive clock switching is performed.

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the Clock sources switch-on process has completed if the target system clock source is one of the following:
  - the 16 MHz internal RC oscillator

- the external crystal oscillator
- the primary PLL

- the Peripheral clocks disable process has completed in order not to change the system clock frequency before peripherals close their internal activitiesthe Peripheral clocks enable process has completed

An overview of system clock source selection possibilities for each mode is shown in the following table. A '√' indicates that a given clock source is selectable for a given mode.

Table 65-3. MC_ME System Clock Selection Overview

System Clock Source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
16 MHz int. RC osc.	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)
external crystal osc.		√		√	√	√	√
primary PLL (PHI)		√		√	√	√	√
secondary PLL		√		√	√		
system clock is disabled		√[1]					

1. disabling the system clock during TEST mode will require a reset in order to exit TEST mode

65.4.3.14 Pad switch-off

If the PDO bit of the ME_<target mode>_MC register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST

This step is executed only after the Peripheral clocks disable process has completed.

65.4.3.15 Clock sources switch-off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to '0'. The following clock sources switched off at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

For the clock sources related to the system clock, this step is executed only after the System clock switching process has completed.

- the 16 MHz internal RC oscillator

65.4.3.16 Flash switch-off

Based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the S_FLA bit field of the ME_GS register. This step is executed only when the Processor Clocks Disable and System clock disable processes have completed.

65.4.3.17 Current mode update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note

SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

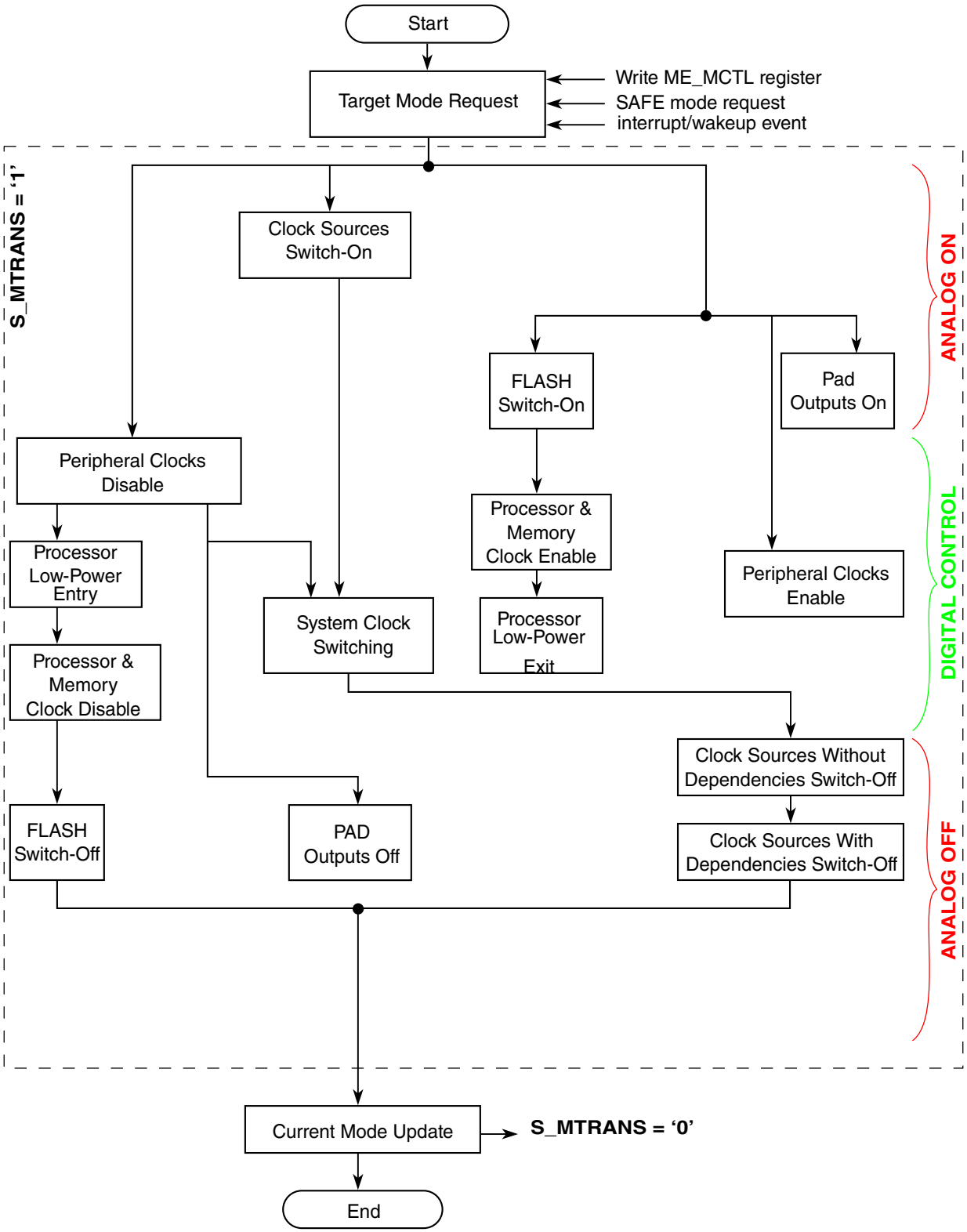


Figure 65-3. MC_ME Transition Diagram

65.4.4 Protection of mode configuration registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz int. RC osc. is selected as the system clock, IRC must be on.
- If the external crystal osc. clock is selected as the system clock, XOSC must be on.
- If the primary PLL (PHI) clock is selected as the system clock, PLL0 must be on.
- If the secondary PLL clock is selected as the system clock, PLL1 must be on.

Note

Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration "00" for the FLAON bit field is reserved.
- System clock configurations marked as 'reserved' may not be selected.
- Configuration "1111" for the SYSCLK bit field is allowed only for theTEST mode, and only in this case may all system clock sources be turned off.

warning

If the system clock is stopped during TEST mode, the chip can exit only via a system reset.

65.4.5 Mode transition interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

65.4.5.1 Invalid mode configuration interrupt

Whenever a write operation is attempted to the ME_<mode>_MC registers violating the protection rules mentioned in the Protection of mode configuration registers, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is '1'.

In addition, during a mode transition, if a clock source has been configured in the ME_<target mode>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0…7/ME_LP_PC0…7 and ME_PCTLn registers, the interrupt pending bit I_ICONF_CU of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CU of the ME_IM register is '1'.

If an attempt to write to one of the ME_CADDRn registers is made after a mode change request has been made via the second write to the ME_MCTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is '1'.

65.4.5.2 Invalid mode transition interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.
- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.

- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition is in progress (the S_MTRANS bit of the ME_GS register is '1'), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.
- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0…3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.
- Self-transition requests (e.g., RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (i.e., S_MTRANS is '1'). During the low-power mode exit process, if the system is not able to enter the respective RUN0…3 mode properly (i.e., all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is '1'.

65.4.5.3 SAFE mode transition interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is '1' .

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (i.e., programming of ME_MCTL register).

65.4.5.4 Mode transition complete interrupt

Whenever the system fully completes a mode transition (i.e., the S_MTRANS bit of ME_GS register transits from '1' to '0'), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is '1'. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

65.4.6 Peripheral clock gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0…7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0…3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTLn registers.

The low-power peripheral configuration registers ME_LP_PC0…7 are chosen only during the low-power modes HALT0 and STOP0. All configurations are programmable by software according to the needs of the application. Each configuration register

contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTLn registers.

Any modifications to the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTLn register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits.

65.4.7 Application example

The following figure shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

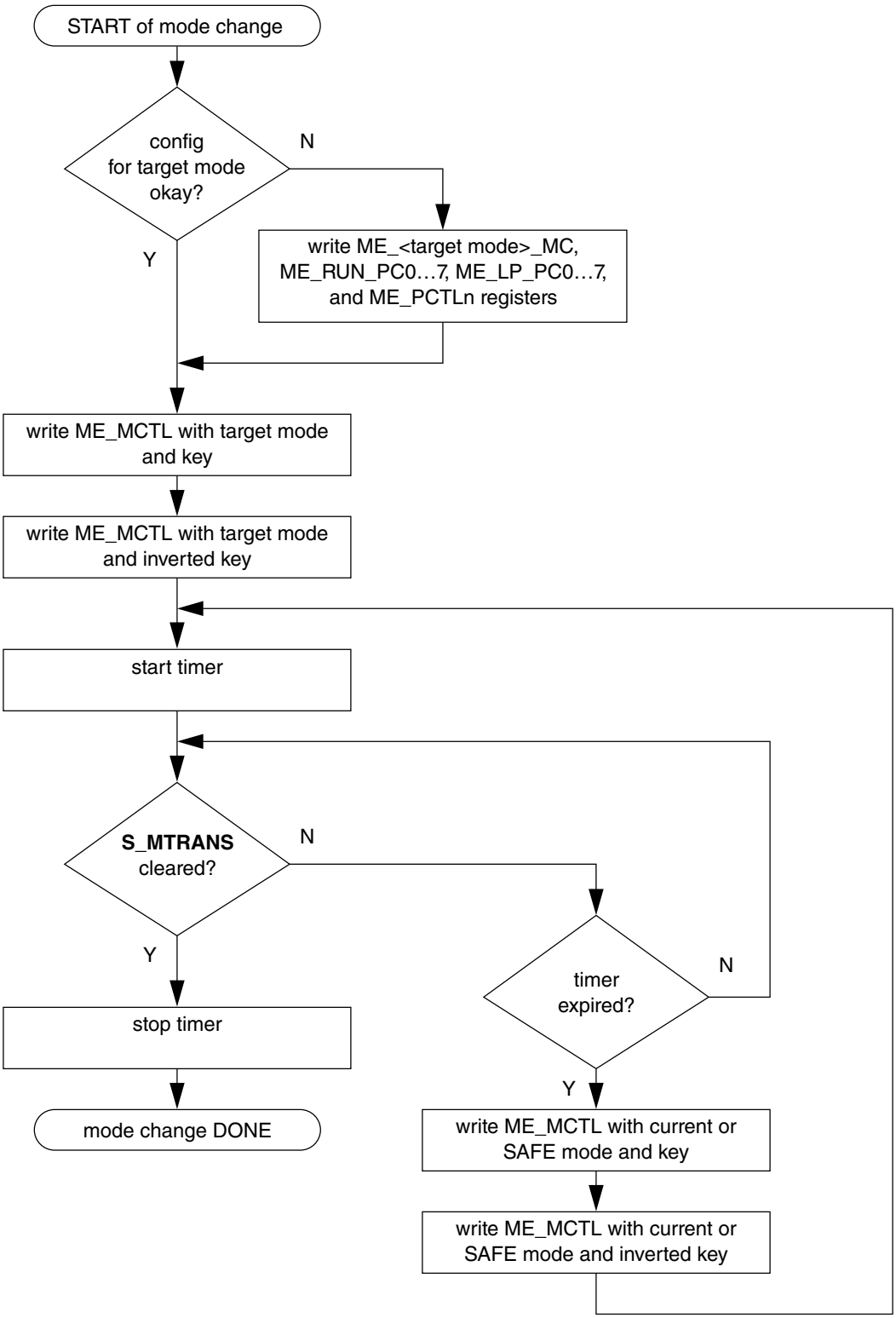


Figure 65-4. MC_ME Application Example Flow Diagram

Chapter 66
Core Debug Support

66.1 Overview

Internal debug support in the e200z425Bn3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. e200z425Bn3 supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, e200z425Bn3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

66.1.1 Software Debug Facilities

e200z425Bn3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

66.1.1.1 *PowerISA 2.06* Compatibility

The e200z425Bn3 core implements a subset of the *PowerISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

66.1.2 Additional Debug Facilities

In addition to the debug functionality defined in *PowerISA 2.06*, e200z425Bn3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

66.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z425Bn3 core. This port provides information corresponding to each read or write access

completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z425Bn3) Nexus 3 Module chapter. The data trace port will report aligned accesses, and most misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0. Also, for accesses generated by a LSP load/store instruction using circular addressing mode which crosses a buffer boundary, the two portions of the misaligned access will be reported separately, since otherwise it is not possible to correctly correlate the data with the initial access address.

66.1.3 Hardware Debug Facilities

The e200z425Bn3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z425Bn3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

66.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

66.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z425Bn3 debug resources.

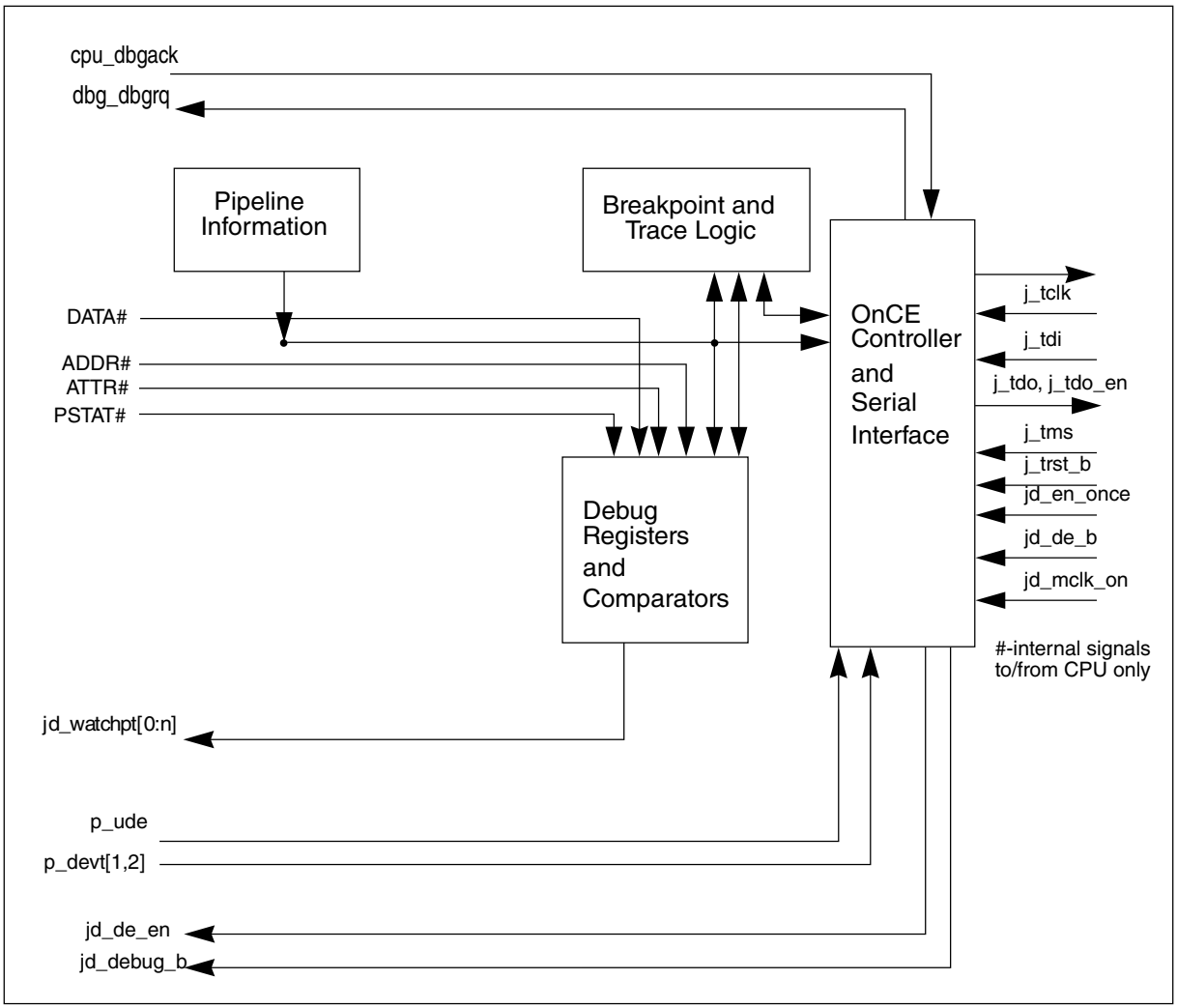


Figure 66-1. e200z425Bn3 Debug Resources

66.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z425Bn3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z425Bn3 debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

66.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

66.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load or store operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the *PowerISA 2.06* definition, Data Address Compare events on e200z425Bn3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC\_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer

however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

66.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC\_OFST}$ field when the DVC DAC status is recorded.

Table 66-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in Table 66-1, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 66-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 66-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 66-2 through Table 66-5 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 66-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.

Table continues on the next page...

Table 66-2. DAC events and Resultant Updates, Dual-issue case 1 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5):	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 66-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	DACy or DVC DACy	

Table 66-3. DAC events and Resultant Updates, Dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3):	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6).

Table continues on the next page...

Table 66-3. DAC events and Resultant Updates, Dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	DACy	**Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACyInstruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 66-4. DAC events and Resultant Updates, Dual-issue case 3

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) alu** **(4) alu** **(5) load/store** **(6) alu**	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 66-5. DAC events and Resultant Updates, Dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

66.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1

may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

66.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

66.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken

debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

66.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

66.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

66.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

66.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

66.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

66.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

66.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

66.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

66.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

66.3.1 Debug Address and Value Registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are

not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z425Bn3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

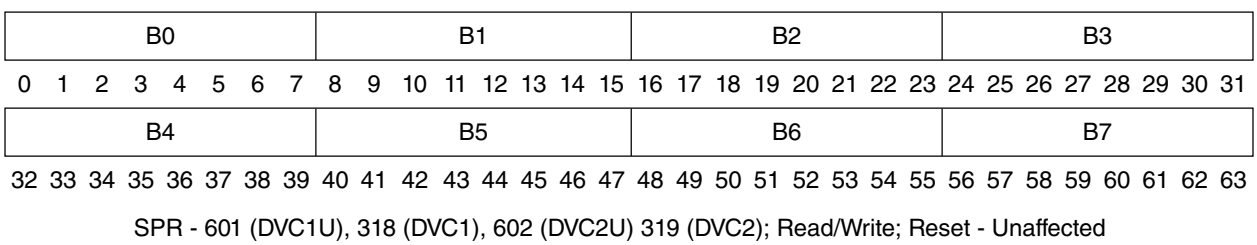


Figure 66-2. DVC1, DVC2 registers

66.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z425Bn3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

66.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z425Bn3 adds some implementation specific bits to this register, as seen in the following figure.

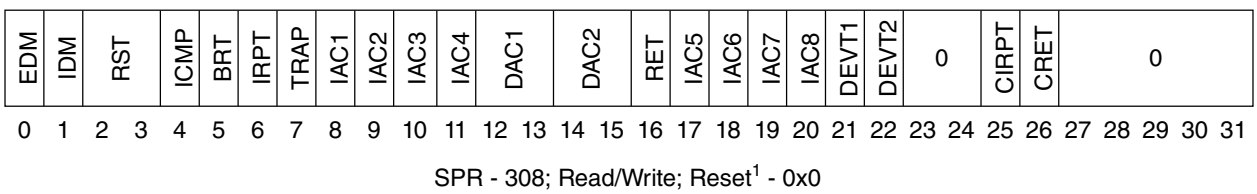


Figure 66-3. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 66-6. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled

Table continues on the next page...

Table 66-6. DBCR0 field descriptions (continued)

Bit	Name	Description
		1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled

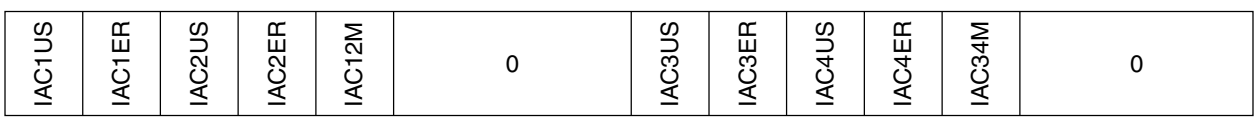
Table continues on the next page...

Table 66-6. DBCR0 field descriptions (continued)

Bit	Name	Description
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

66.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.



SPR - 309; Read/Write; Reset[1] - 0x0

Figure 66-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 66-7. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode

Table continues on the next page...

Table 66-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		00 IAC1 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode
		00 IAC1 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode
		00 IAC2 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode
		00 IAC2 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode
		00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2.
		01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode
		00 IAC3 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode
		00 IAC3 debug events are based on effective address

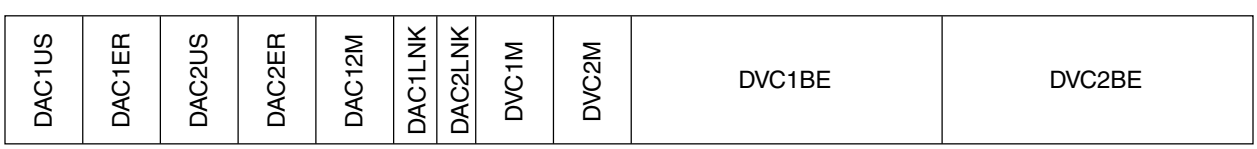
Table continues on the next page...

Table 66-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

66.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.



SPR - 310; Read/Write; Reset[1] - 0x0

Figure 66-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 66-8. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table continues on the next page...

Table 66-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access.

Table continues on the next page...

Table 66-8. DBCR2 Bit Definitions (continued)

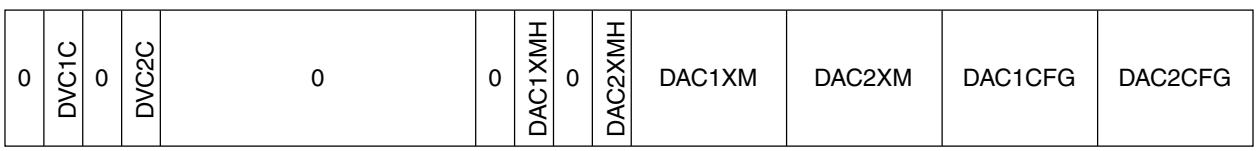
Bit	Name	Description
		10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2.

Table 66-8. DBCR2 Bit Definitions

Bit	Name	Description
		xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

66.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 66-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 66-9. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation.

Table continues on the next page...

Table 66-9. DBCR4 field description (continued)

Bit	Name	Description
		1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 - DAC1XM masks 0–15 low-order address bits 1 - DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).

Table continues on the next page...

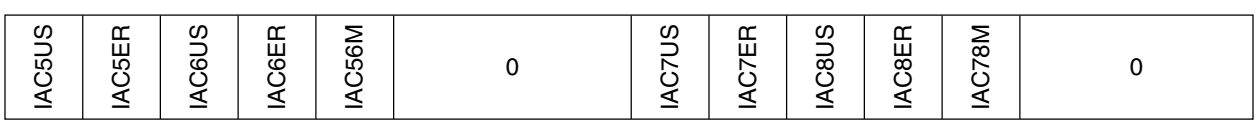
Table 66-9. DBCR4 field description (continued)

Bit	Name	Description
		NOTE: unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

66.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 66-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 66-10. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved

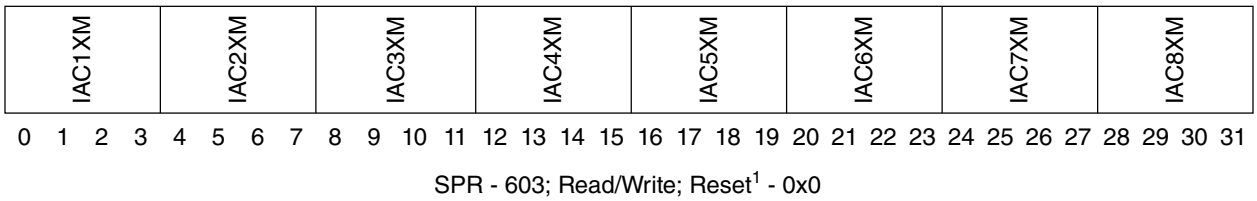
Table continues on the next page...

Table 66-10. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

66.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.



SPR - 603; Read/Write; Reset[1] - 0x0

Figure 66-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 66-11. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported.

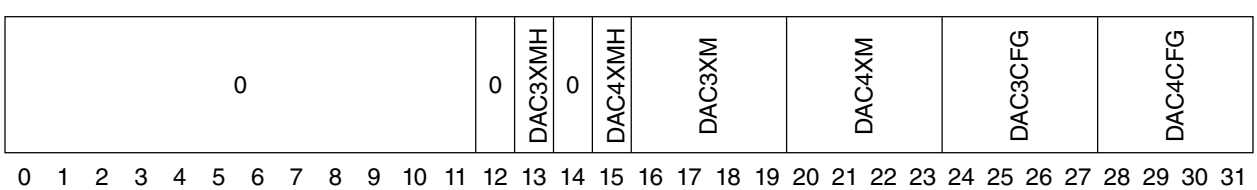
Table continues on the next page...

Table 66-11. DBCR6 field descriptions (continued)

Bit	Name	Description
		1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

66.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.



SPR - 596; Read/Write; Reset[1] - 0x0

Figure 66-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 66-12. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses

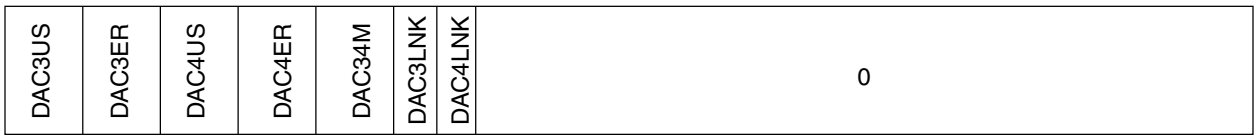
Table continues on the next page...

Table 66-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

66.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 66-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 66-13. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0

Table continues on the next page...

Table 66-13. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

66.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no

effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.

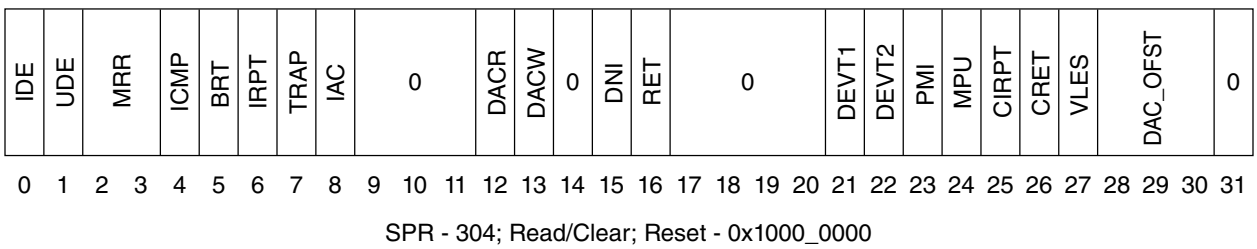


Figure 66-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 66-14. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event

Table continues on the next page...

Table 66-14. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

66.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load or store operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

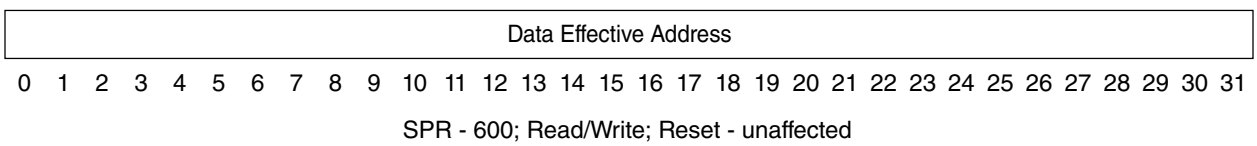


Figure 66-12. DDEAR Register

66.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

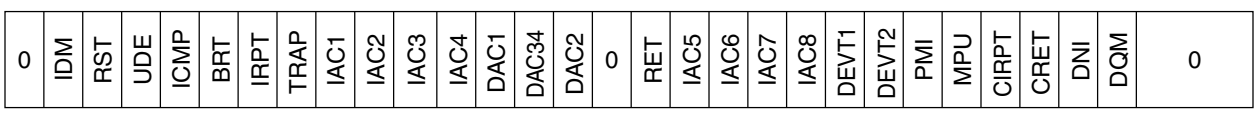
The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.



SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 66-13. EDBRAC0 Register

Table 66-14 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 66-15. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.

Table continues on the next page...

Table 66-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields.

Table continues on the next page...

Table 66-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event

Table continues on the next page...

Table 66-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 - Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 - Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** this bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 66-16 shows which resources are controlled by EDBRAC0 settings.

Table 66-16. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$,

Table continues on the next page...

Table 66-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
																												$DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table continues on the next page...

Table 66-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	**Software Accessible via mtspr,affected by p_reset_b**
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	$DBCR0_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	$DBCR0_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	$DBCR0_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN,DWDEN,IDEN}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	$DBCR0_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	$DBCR0_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 66-16 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

66.3.4 Debug Event Select Register (DEVENT)

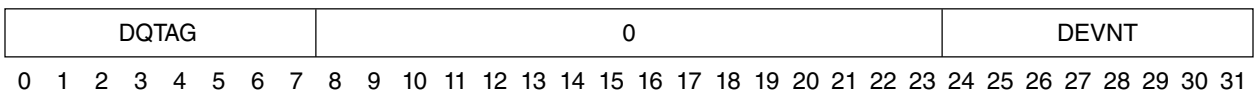
The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has

occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z425Bn3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.



SPR - 975; Reset[1] - 0x0

Figure 66-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 66-17. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx - **p_devnt_out[4]** is asserted for one clock xx1xxxxx - **p_devnt_out[5]** is asserted for one clock

Table 66-17. DEVENT Bit Definitions

Bit(s)	Name	Description
		x1xxxxxx - **p_devnt_out[6]** is asserted for one clock 1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

66.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z425Bn3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

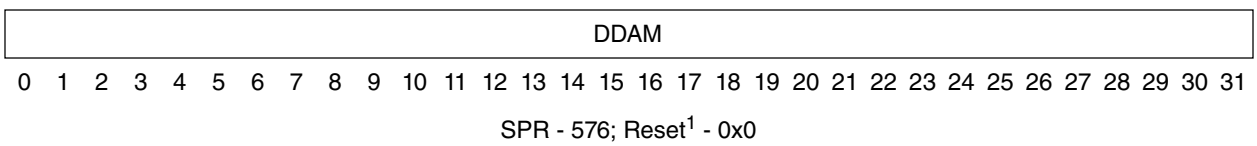


Figure 66-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 66-18. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

66.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 66-9. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses,

an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 66-9 and Table 66-12 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

66.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is

performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

66.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

66.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

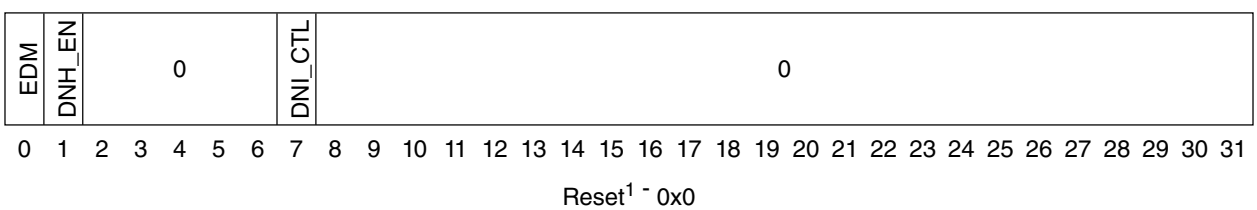


Figure 66-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

The following table provides bit definitions for External Debug Control Register 0.

Table 66-19. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

66.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0
0	1	2	3	4	5	6	7	8	9 10 11	12	13	14	15	16	17 18 19 20	21	22	23	24	25	26	27	28 29 30	31

Read/Write; Reset[1] - 0x0000_0000

Figure 66-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 66-20. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved

Table continues on the next page...

Table 66-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH\_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

66.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

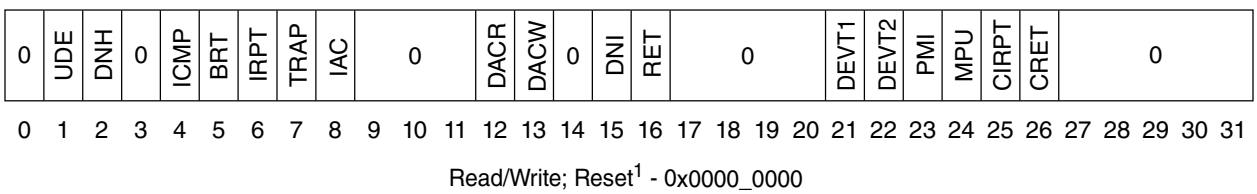


Figure 66-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 66-21. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$

Table continues on the next page...

Table 66-21. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

66.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load or store operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent

hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

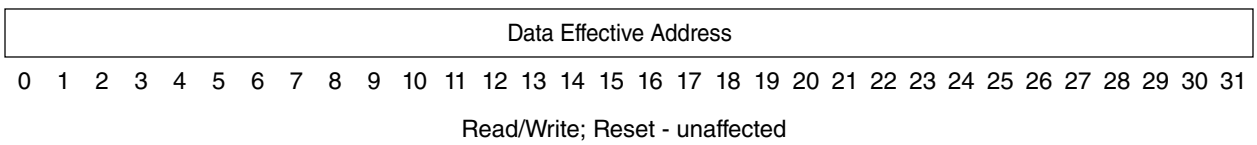


Figure 66-19. EDDEAR Register

66.5.2 OnCE Introduction

The e200z425Bn3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z425Bn3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/ software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

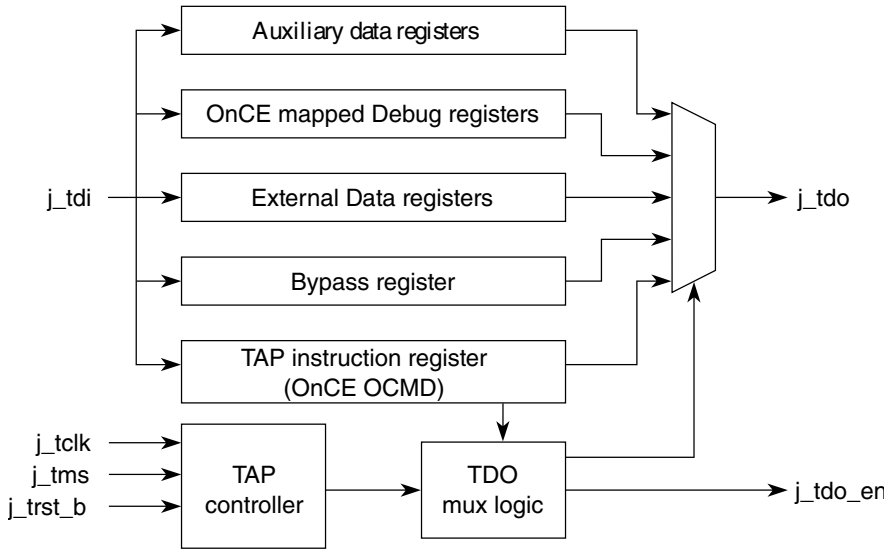
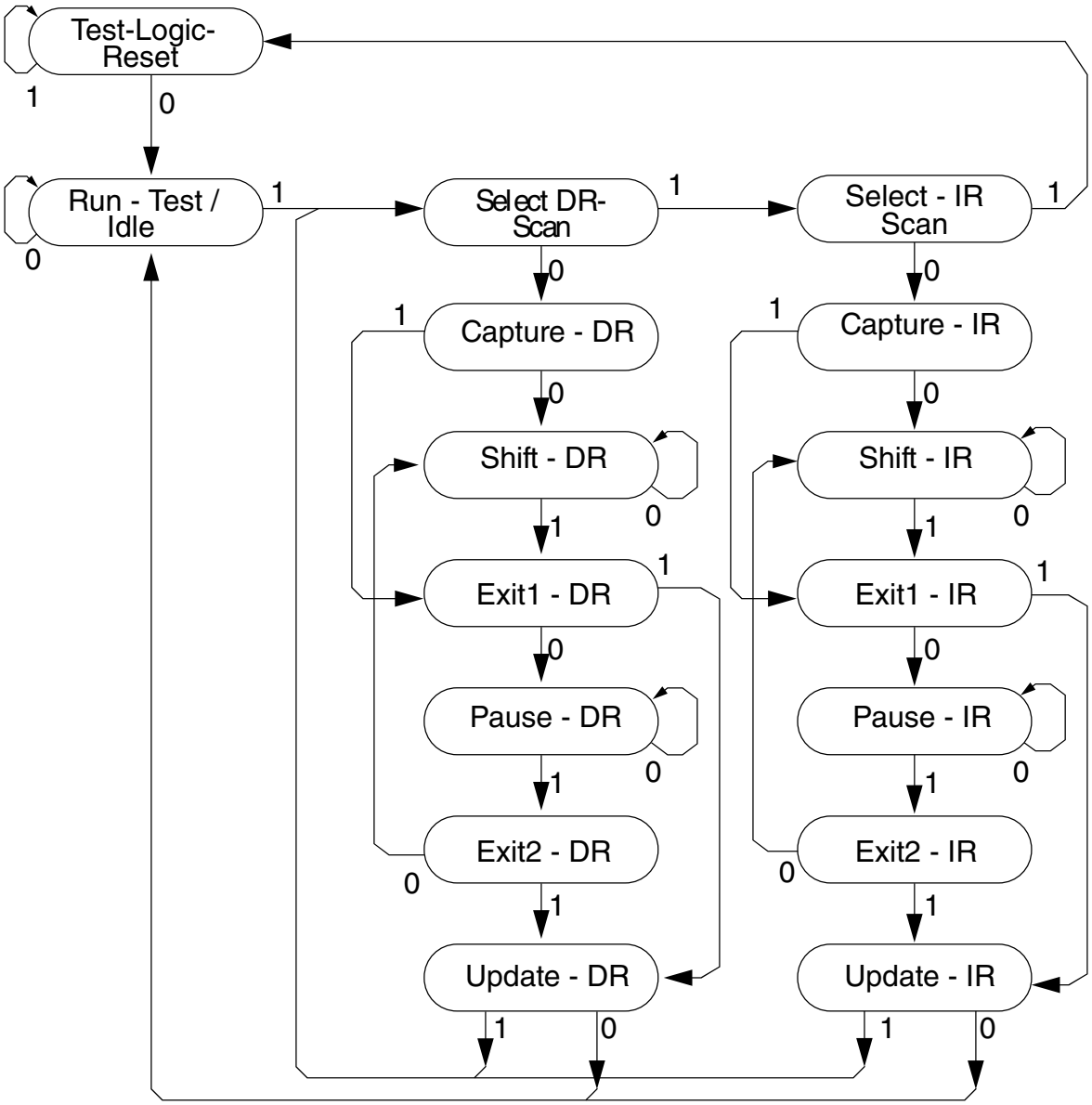


Figure 66-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

66.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 66-22. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z425Bn3) Core Complex Overview.

66.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

66.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

66.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

66.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

66.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

66.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

66.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE

commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

66.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z425Bn3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

66.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

66.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z425Bn3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

66.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

66.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

66.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

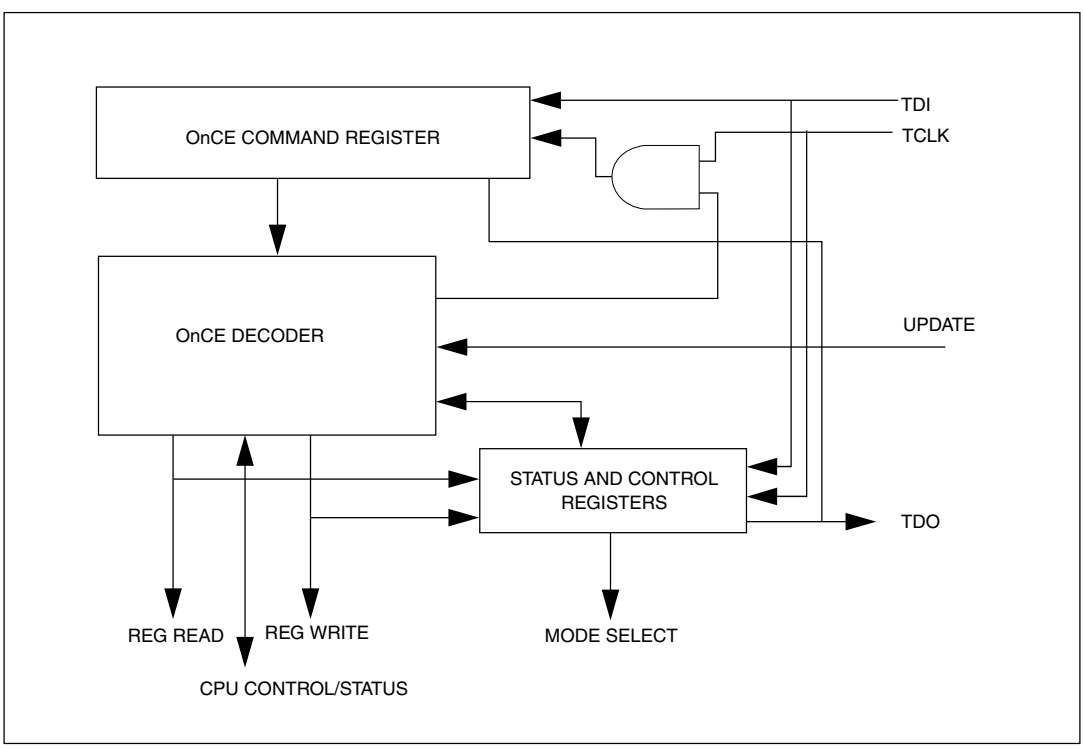


Figure 66-21. OnCE Controller and Serial Interface

66.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 66-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 66-23. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.

Table continues on the next page...

Table 66-23. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

66.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 66-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 66-23. OnCE Command Register

Table 66-24 provides bit definitions for the Once Command Register.

Table 66-24. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6]

Table continues on the next page...

Table 66-24. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 66-25 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 66-25 indicates the OnCE register addresses.

Table 66-25. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved

Table continues on the next page...

Table 66-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)

Table continues on the next page...

Table 66-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Reserved for Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

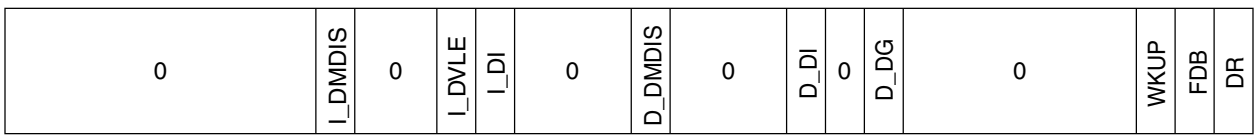
Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

66.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z425Bn3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see MPU Operation During Debug). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.



Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 66-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 66-26. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit (I_DVLE) This bit is used to provide the 'VLE' attribute bit to be used during a debug session. Note: this bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug 'I' Attribute Bit (I_DI) This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions

Table continues on the next page...

Table 66-26. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
		This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

66.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z425Bn3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z425Bn3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z425Bn3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 66-27. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*	
DBCR0	Y	Y	Y	N	*[1]	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	

Table continues on the next page...

Table 66-27. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.

66.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

66.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z425Bn3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z425Bn3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

66.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

66.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

66.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled

along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

66.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z425Bn3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

66.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH\_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z425Bn3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

66.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

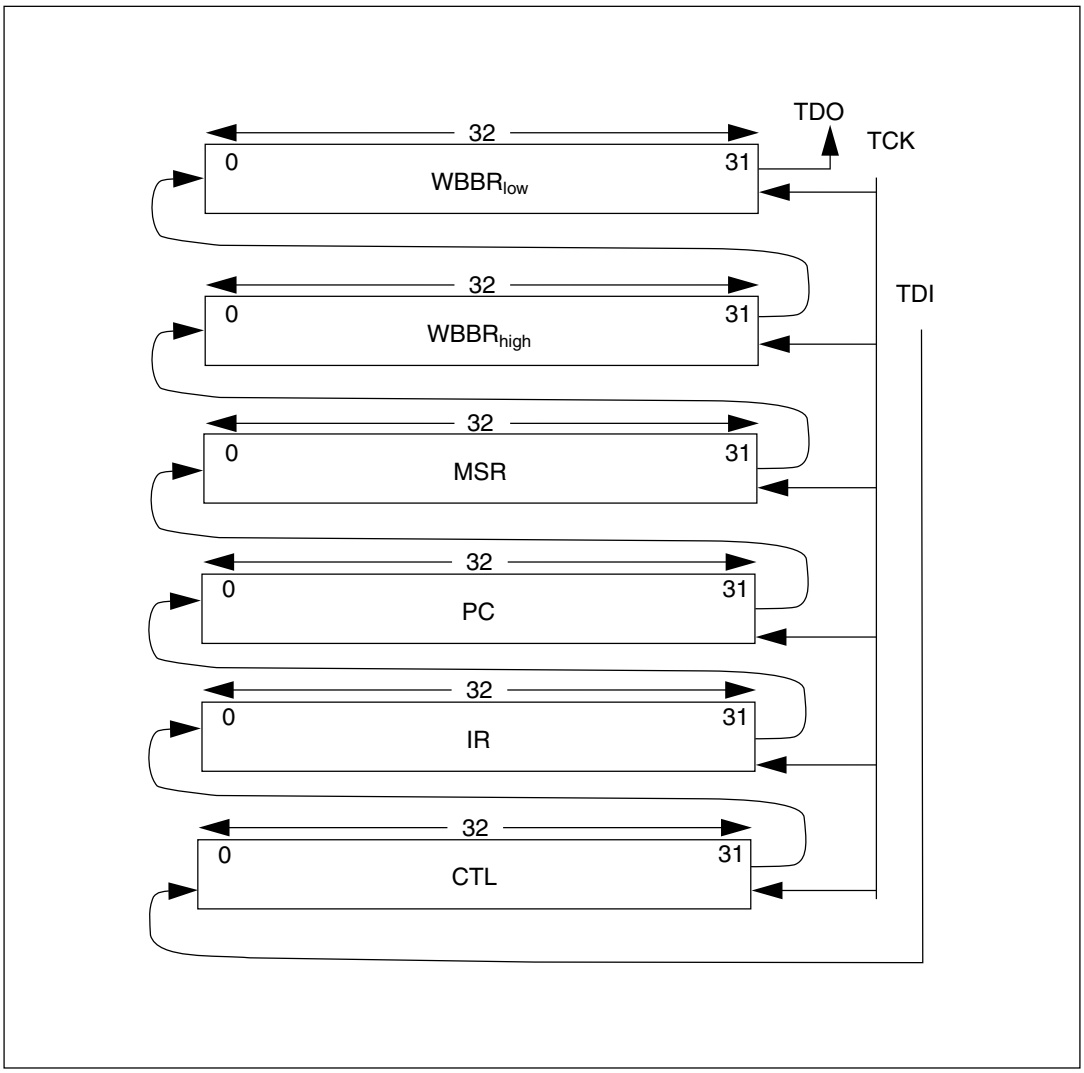


Figure 66-25. CPU Scan Chain Register (CPUSCR)

66.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug

control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

66.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

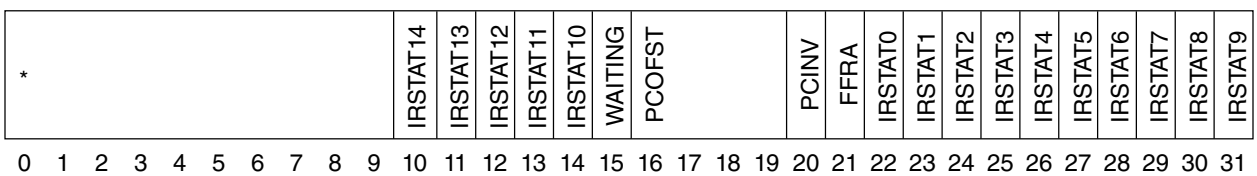


Figure 66-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.For most LSP instructions using rA||rB as a 64-bit source operand, $WBBR_{hi, lo}$ is used to supply the 64-bit source value.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$or $WBBR_{hi,lo}$used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the PowerISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0= No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1= An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be

written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

66.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

66.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. For LSP instructions which generate 64-bit results, $WBBR_{low}$ holds the low-order 32 bits of the result. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. For LSP instructions which generate 64-bit results, $WBBR_{high}$ holds the high-order 32 bits of the result. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

66.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

66.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction

following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

66.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

66.6 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

66.7 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

66.8 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode.
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has re-entered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should re-write the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 67
Core Debug Support

67.1 Overview

Internal debug support in the e200z710n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *Power ISA 2.06*. e200z710n3 supports a subset of these defined facilities. In addition to the facilities defined in *Power ISA 2.06*, e200z710n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

67.1.1 Software Debug Facilities

e200z710n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

67.1.1.1 *Power ISA 2.06* Compatibility

The e200z710n3 core implements a subset of the *Power ISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

67.1.2 Additional Debug Facilities

In addition to the debug functionality defined in *Power ISA 2.06*, e200z710n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

67.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z710n3 core. This port provides information corresponding to each read or write access

completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z710n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

67.1.3 Hardware Debug Facilities

The e200z710n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard IEEE 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z710n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

67.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

67.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z710n3 debug resources.

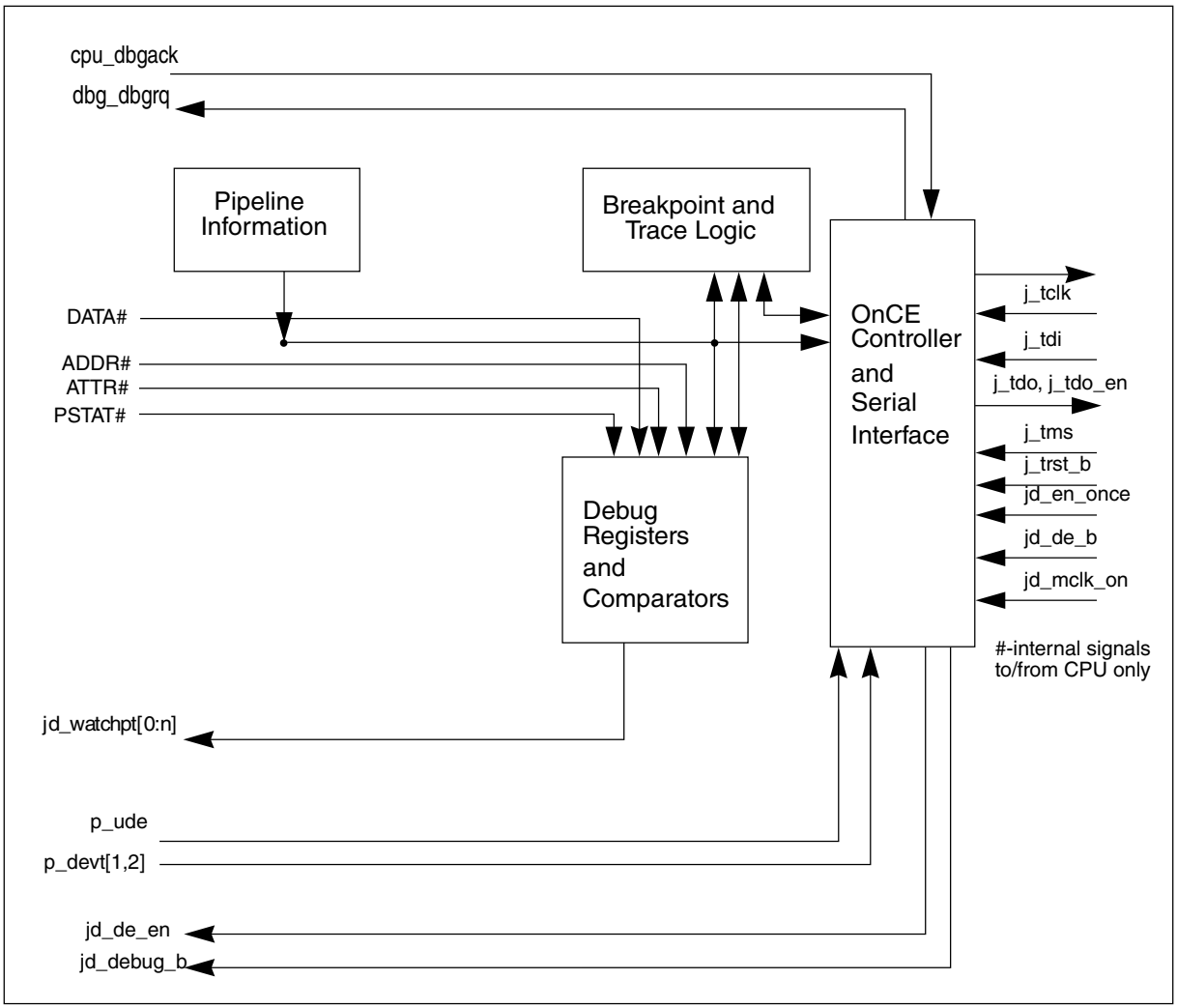


Figure 67-1. e200z710n3 Debug Resources

67.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *Power ISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z710n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z710n3 debug configuration supports most of these event types. Unsupported *Power ISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

67.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

67.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the *Power ISA 2.06* definition, Data Address Compare events on e200z710n3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC\_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, i.e. no data comparisons are performed since these instructions do not have associated data values.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/ EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

67.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/ store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC\_OFST}$ field when the DVC DAC status is recorded.

Table 67-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to

this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0.

Table 67-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction.

Table continues on the next page...

Table 67-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
			Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

67.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

67.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

67.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

67.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

67.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

67.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

67.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

67.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

67.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

67.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

67.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

67.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction

targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

67.3.1 Debug Address and Value Registers

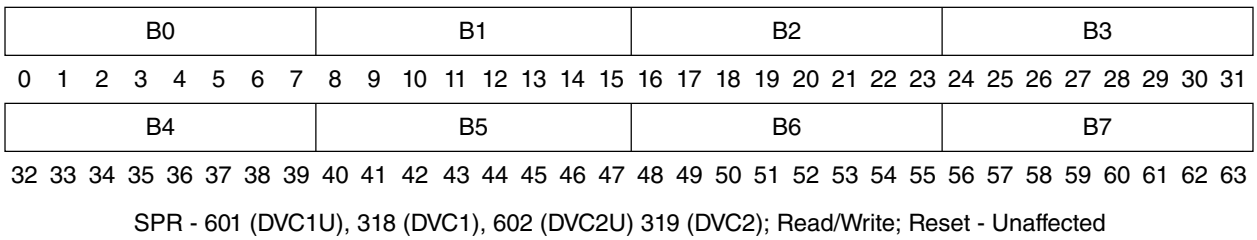
Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z710n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned

access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.



SPR - 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset - Unaffected

Figure 67-2. DVC1, DVC2 registers

67.3.2 Debug Control and Status registers

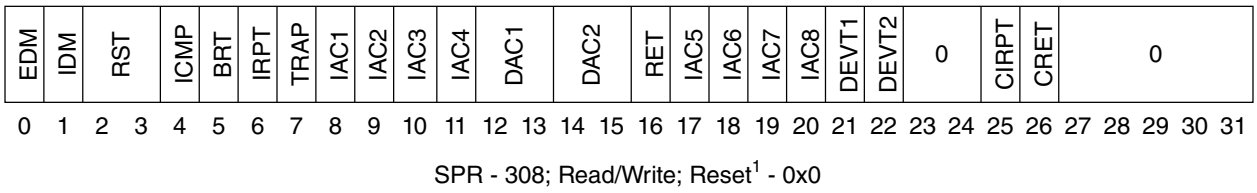
Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z710n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

67.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z710n3 adds some implementation specific bits to this register, as seen in the following figure.



SPR - 308; Read/Write; Reset[1] - 0x0

Figure 67-3. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 67-2. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control

Table continues on the next page...

Table 67-2. DBCR0 field descriptions (continued)

Bit	Name	Description
		00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses

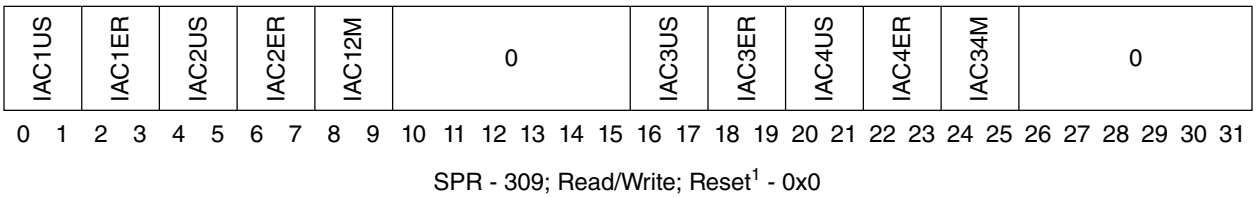
Table continues on the next page...

Table 67-2. DBCR0 field descriptions (continued)

Bit	Name	Description
		11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

67.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.



SPR - 309; Read/Write; Reset[1] - 0x0

Figure 67-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 67-3. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode

Table continues on the next page...

Table 67-3. DBCR1 field descriptions (continued)

Bit	Name	Description
		00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.

Table continues on the next page...

Table 67-3. DBCR1 field descriptions (continued)

Bit	Name	Description
		10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

67.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.

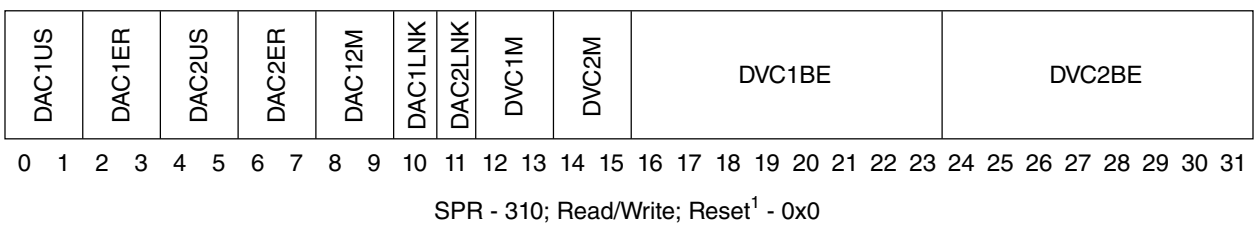


Figure 67-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 67-4. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode

Table continues on the next page...

Table 67-4. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0:

Table continues on the next page...

Table 67-4. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables

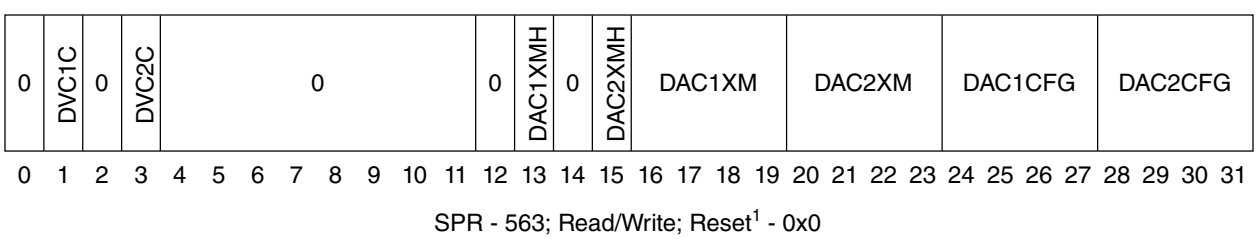
Table continues on the next page...

Table 67-4. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

67.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 67-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 67-5. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM:

Table continues on the next page...

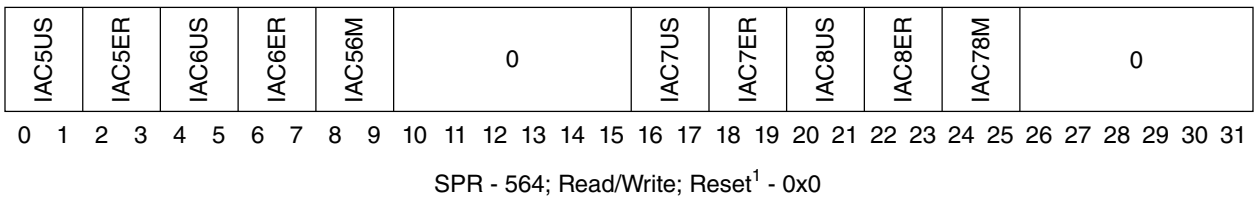
Table 67-5. DBCR4 field description (continued)

Bit	Name	Description
		00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **NOTE:** unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

67.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 67-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 67-6. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)

Table continues on the next page...

Table 67-6. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0

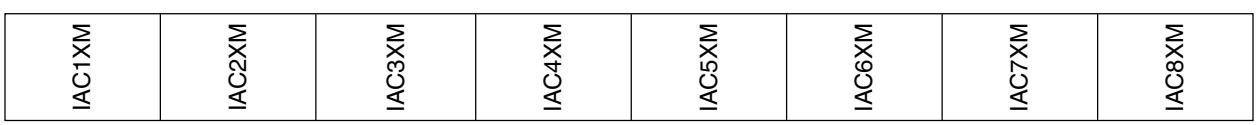
Table continues on the next page...

Table 67-6. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

67.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.



SPR - 603; Read/Write; Reset[1] - 0x0

Figure 67-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 67-7. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8.

Table continues on the next page...

Table 67-7. DBCR6 field descriptions (continued)

Bit	Name	Description
		0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

67.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.

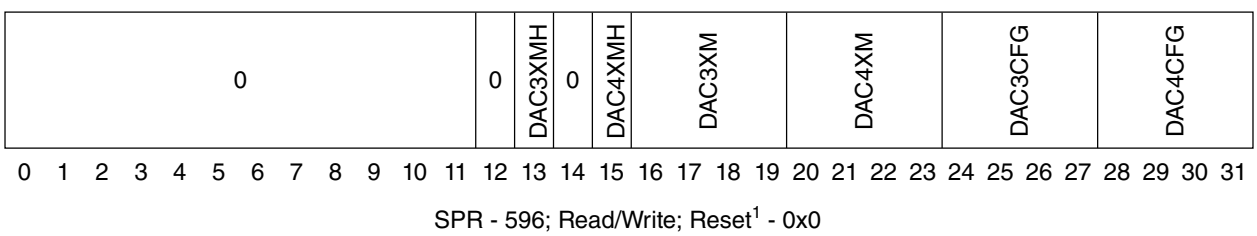


Figure 67-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 67-8. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved

Table continues on the next page...

Table 67-8. DBCR7 Field Descriptions (continued)

Bit	Name	Description
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses

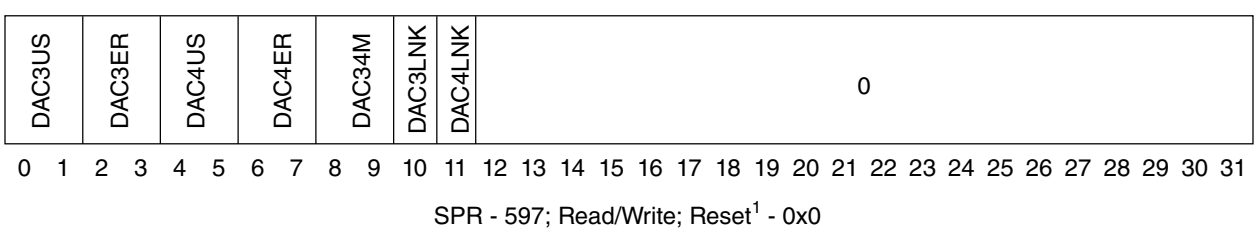
Table continues on the next page...

Table 67-8. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

67.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 67-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 67-9. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4.

Table continues on the next page...

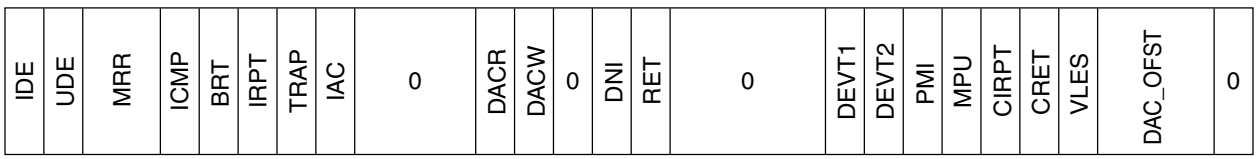
Table 67-9. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

67.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource

sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.



SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 67-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 67-10. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred

Table continues on the next page...

Table 67-10. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

67.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

Data Effective Address																														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 600; Read/Write; Reset - unaffected

Figure 67-12. DDEAR Register

67.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 67-13. EDBRAC0 Register

Table 67-10 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 67-11. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.

Table continues on the next page...

Table 67-11. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields.

Table continues on the next page...

Table 67-11. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event

Table continues on the next page...

Table 67-11. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 - Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 - Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** this bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 67-12 shows which resources are controlled by EDBRAC0 settings.

Table 67-12. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$,

Table continues on the next page...

Table 67-12. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
																												$DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table continues on the next page...

Table 67-12. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	$DBCR0_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	$DBCR0_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	$DBCR0_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN,DWDEN,IDEN}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	$DBCR0_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	$DBCR0_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 67-12 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

67.3.4 Debug Event Select Register (DEVENT)

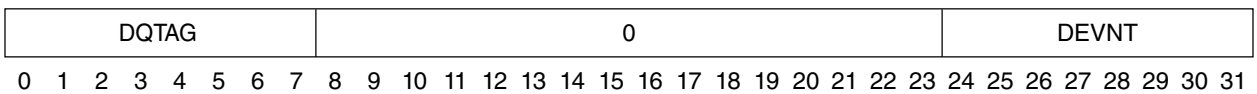
The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has

occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z710n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.



SPR - 975; Reset[1] - 0x0

Figure 67-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 67-13. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx - **p_devnt_out[4]** is asserted for one clock xx1xxxxx - **p_devnt_out[5]** is asserted for one clock

Table 67-13. DEVENT Bit Definitions

Bit(s)	Name	Description
		x1xxxxxx - **p_devnt_out[6]** is asserted for one clock 1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

67.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z710n3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

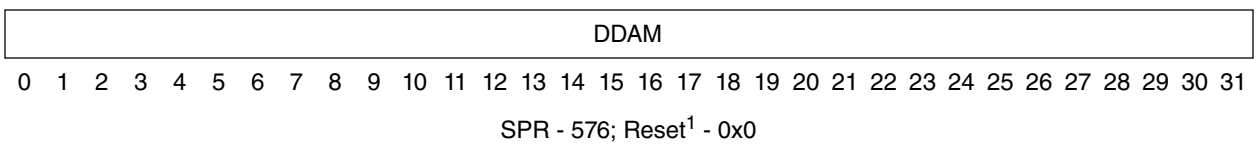


Figure 67-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 67-14. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

67.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 67-5. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0

pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 67-5 and Table 67-8 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

67.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is

performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

67.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

67.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

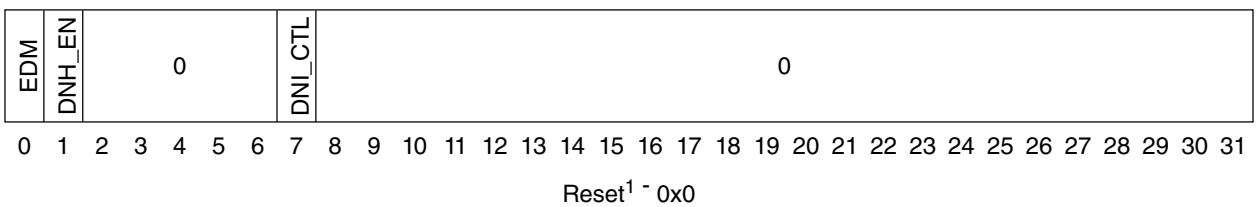


Figure 67-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

The following table provides bit definitions for External Debug Control Register 0.

Table 67-15. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

67.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.

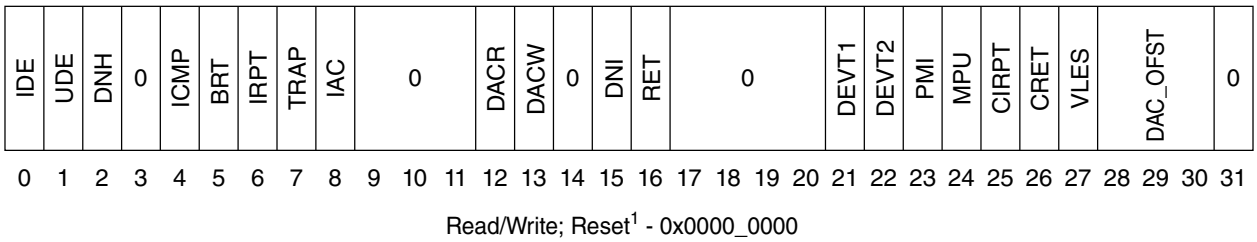


Figure 67-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 67-16. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved

Table continues on the next page...

Table 67-16. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH\_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

67.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

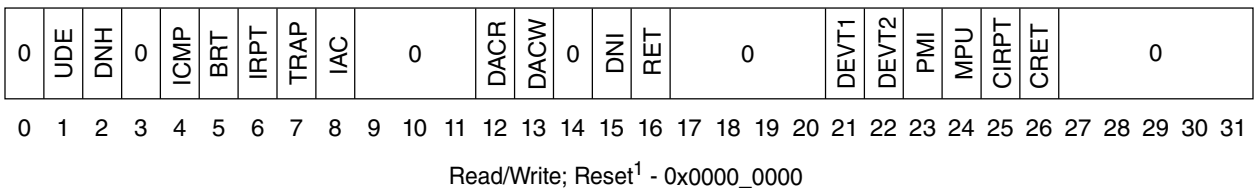


Figure 67-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 67-17. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$

Table continues on the next page...

Table 67-17. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

67.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR

hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

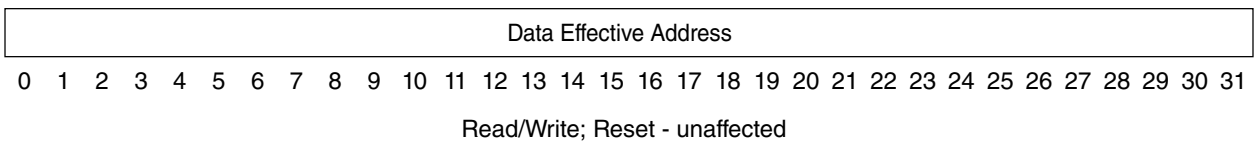


Figure 67-19. EDDEAR Register

67.5.2 OnCE Introduction

The e200z710n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z710n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

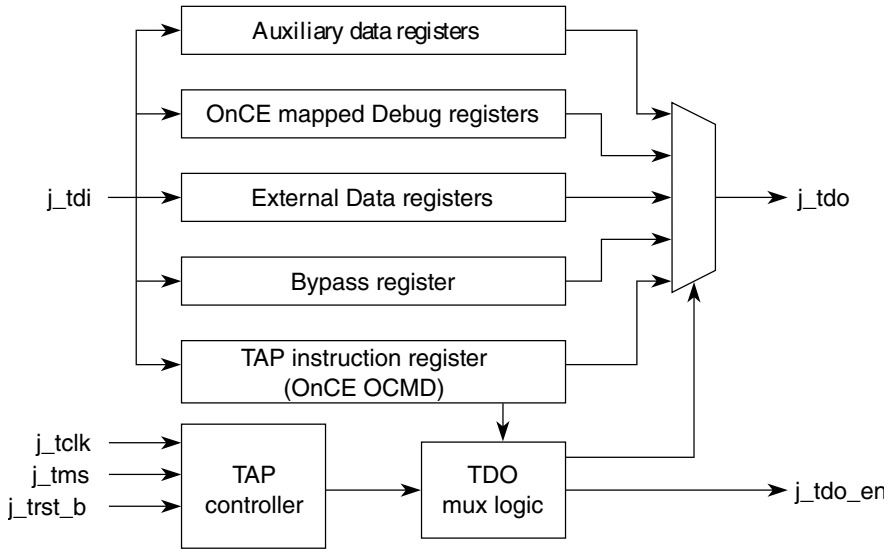
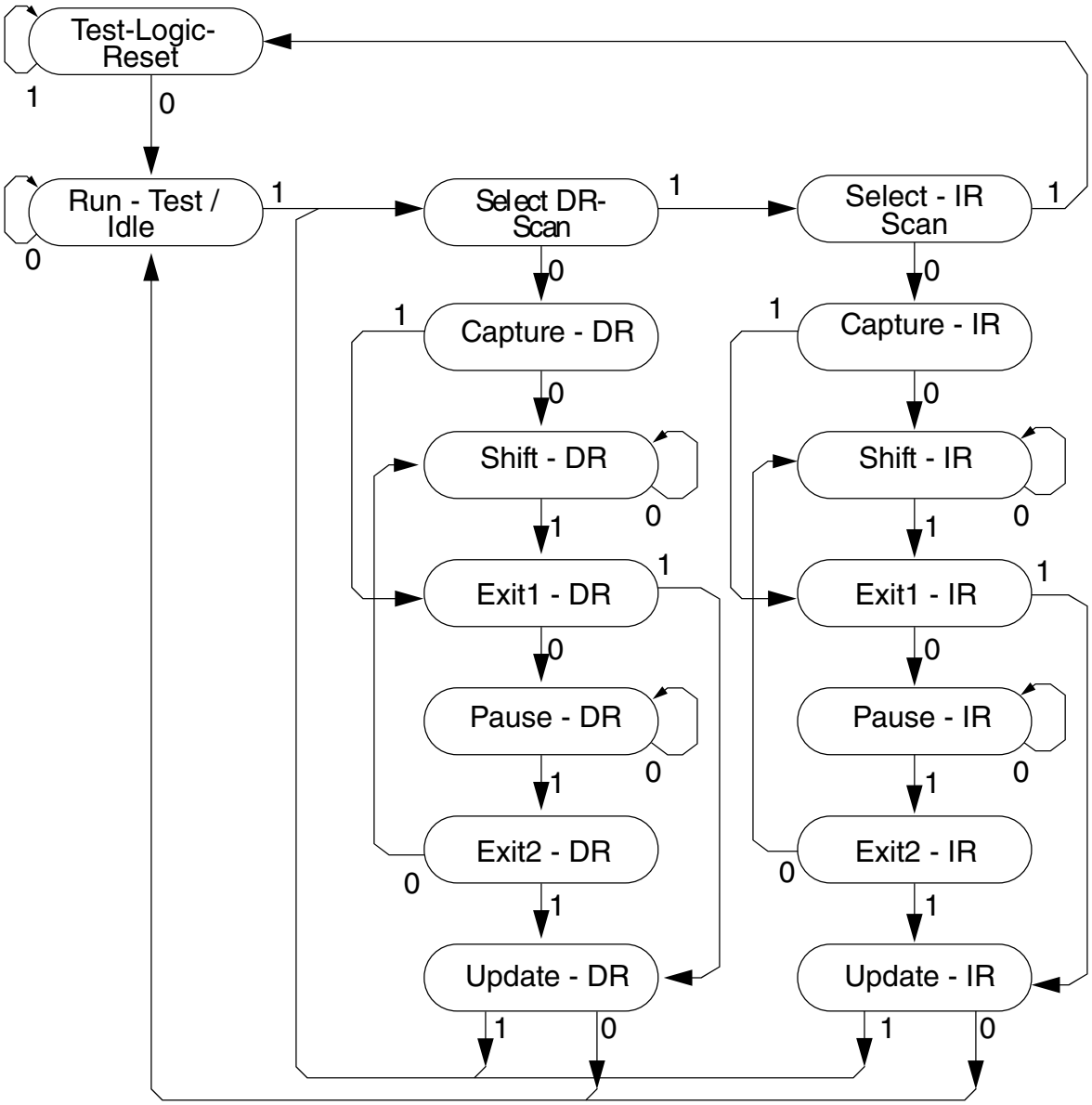


Figure 67-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

67.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 67-18. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z710n3) Core Complex Overview.

67.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

67.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

67.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

67.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

67.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

67.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

67.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE

commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

67.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z710n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

67.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

67.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z710n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

67.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

67.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

67.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

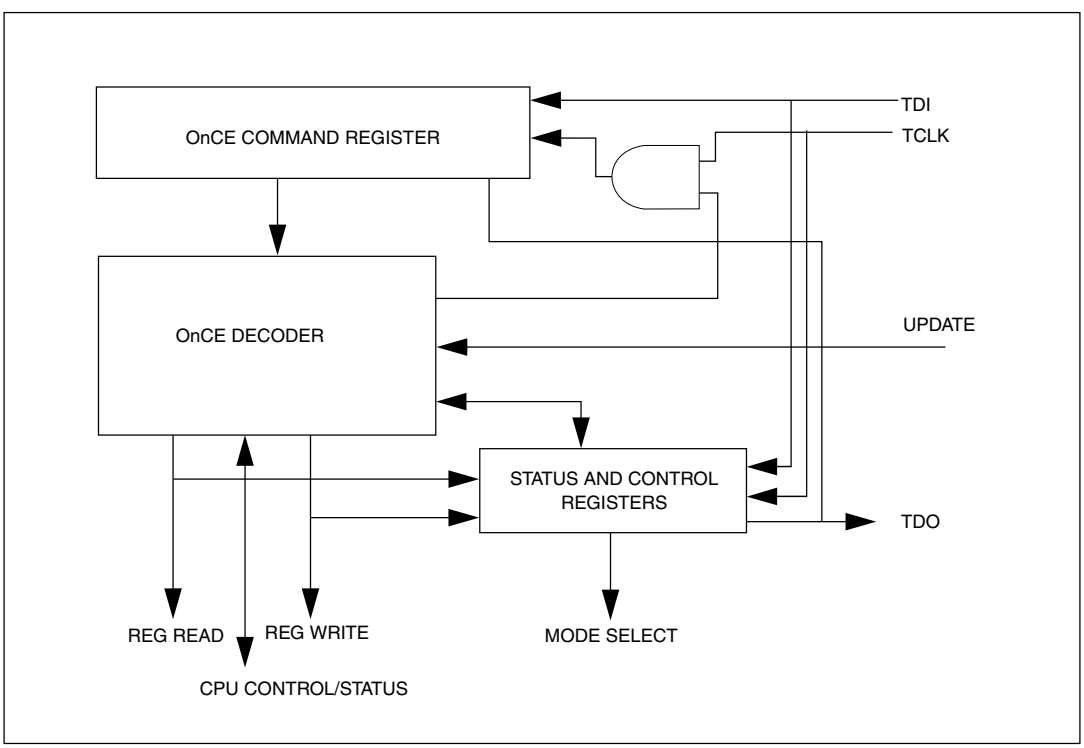


Figure 67-21. OnCE Controller and Serial Interface

67.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 67-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 67-19. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.

Table continues on the next page...

Table 67-19. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

67.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 67-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 67-23. OnCE Command Register

Table 67-20 provides bit definitions for the Once Command Register.

Table 67-20. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6]

Table continues on the next page...

Table 67-20. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 67-21 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 67-21 indicates the OnCE register addresses.

Table 67-21. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved

Table continues on the next page...

Table 67-21. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)

Table continues on the next page...

Table 67-21. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

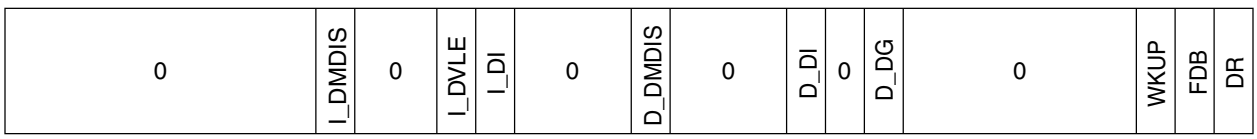
Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

67.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z710n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see MPU Operation During Debug). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.



Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 67-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 67-22. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit (I_DVLE) This bit is used to provide the 'VLE' attribute bit to be used during a debug session. Note: this bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug 'I' Attribute Bit (I_DI) This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions

Table continues on the next page...

Table 67-22. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
		This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

67.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z710n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z710n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z710n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 67-23. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*[1]	
DBCR0	Y	Y	Y	N	*[1]	*$EDBCR0_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N[2]	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	

Table continues on the next page...

Table 67-23. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

67.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

67.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z710n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z710n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

67.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

67.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

67.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled

along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

67.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z710n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

67.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH\_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z710n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

67.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

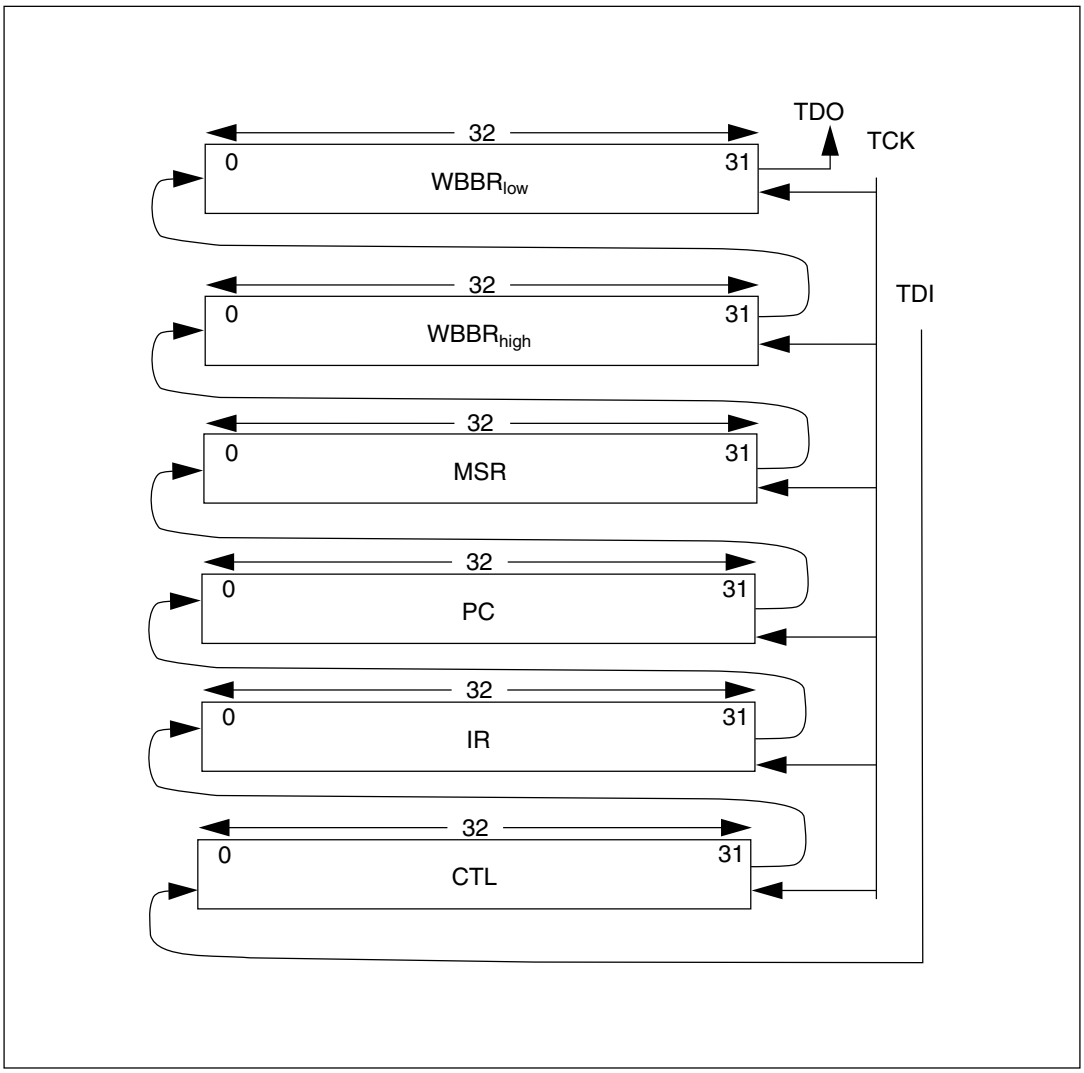


Figure 67-25. CPU Scan Chain Register (CPUSCR)

67.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug

control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

67.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

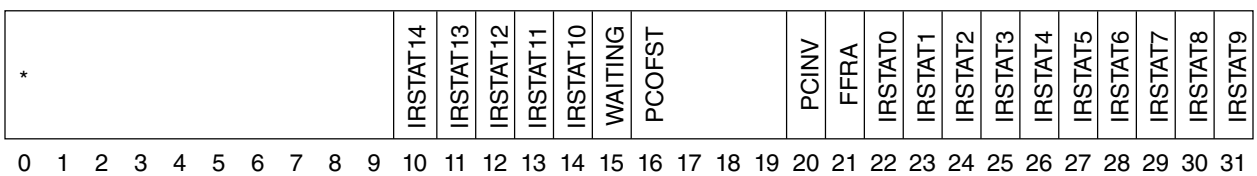


Figure 67-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$ used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the Power ISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a Power ISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0= No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1= An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be

written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

67.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

67.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

67.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

67.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction

following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

67.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

67.6 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

67.7 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

67.8 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode.
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has re-entered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should re-write the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 68
Debug and Calibration Interface (DCI)

68.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard IEEE 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only.

68.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- IEEE 1149.1 and IEEE 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPU(s) when exiting debug mode
- Tool hot plug capability

- Security access control
- Debug reset control

68.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.
    - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST's 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
    - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as timers/ SPU, etc. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

This figure shows the DCI block diagram.

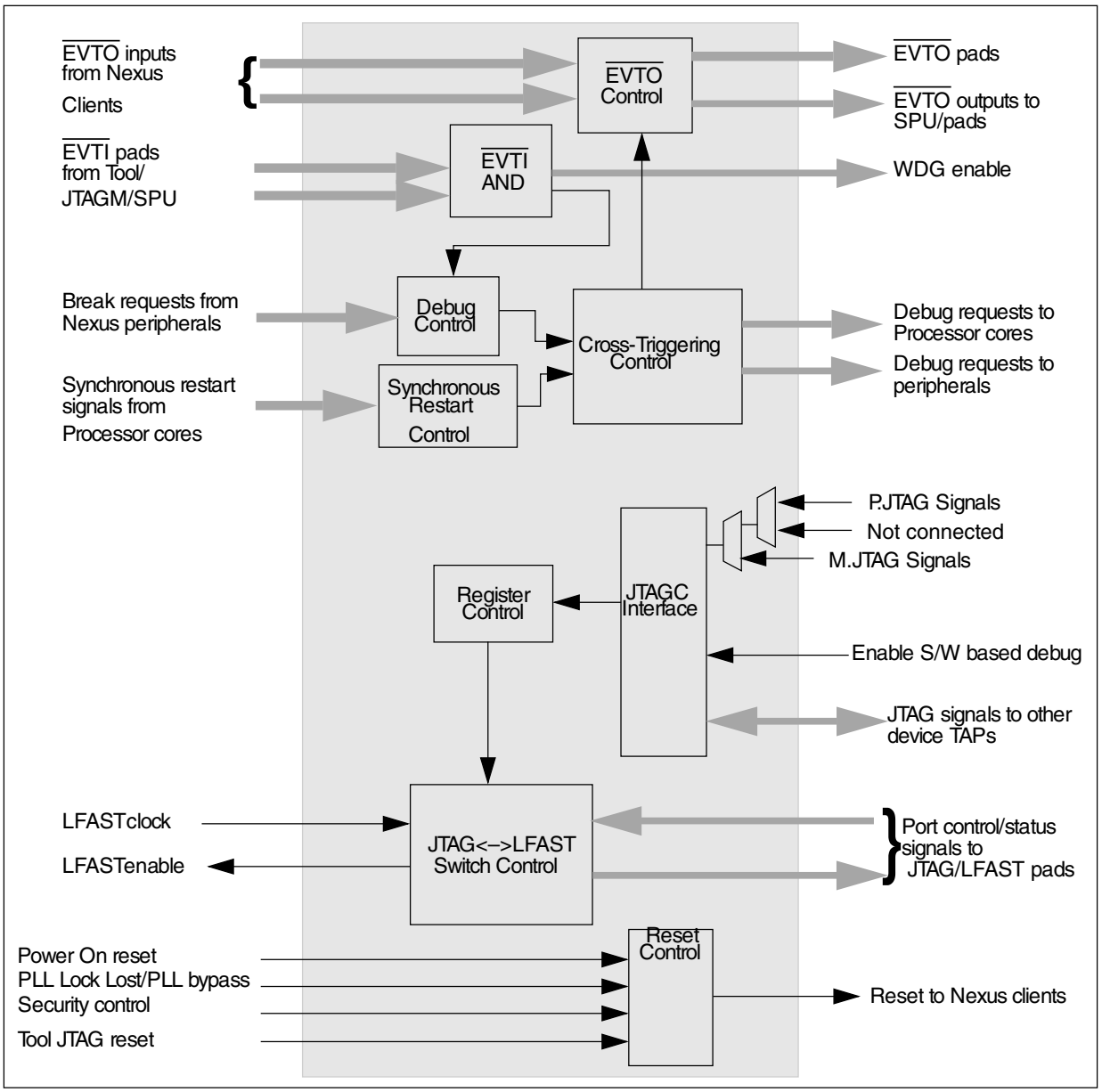


Figure 68-1. DCI block diagram

This figure shows the DCI configuration for the standard production package.

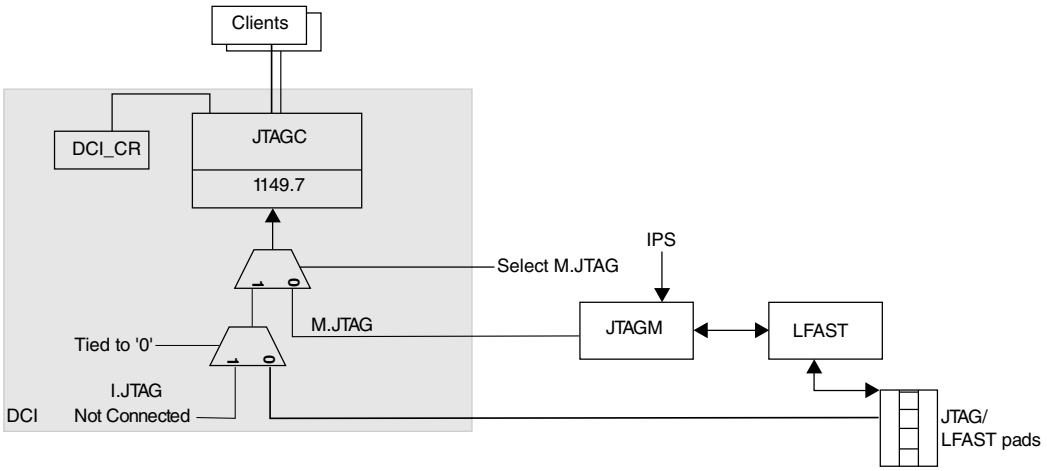


Figure 68-2. DCI in a standard configuration

68.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

This table shows a summary of the configuration options.

Table 68-1. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI Source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the LFAST and JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed. Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in Port sharing between JTAG and LFAST modes. During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. Table 68-2 shows the port mapping.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 68-2. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

This figure shows the DCI mode switching operation.

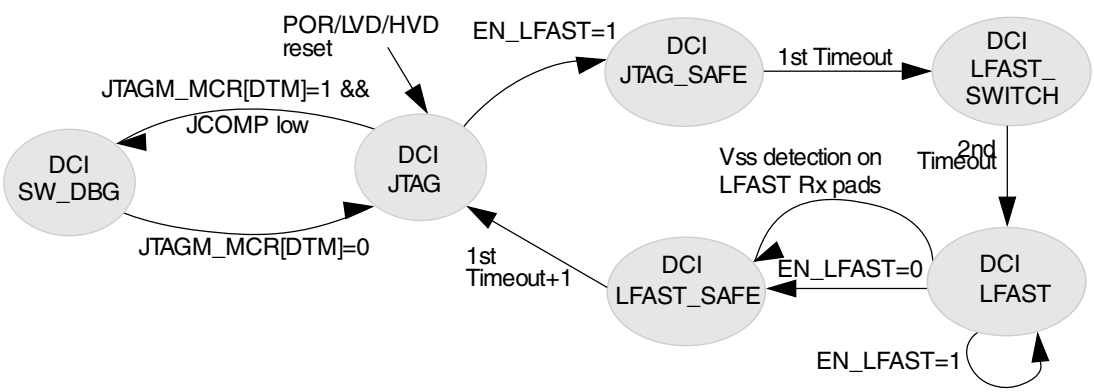


Figure 68-3. DCI operating modes state diagram

Note

JTAGM_MCR[jtagm_JCOMP] should not be asserted in LFAST mode to avoid any undesired behavior.

This table contains the operating mode descriptions.

Table 68-3. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: • JCOMP is latched inside to control until switching gets completed • The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state • An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

This figure shows a block diagram of the JTAG/LFAST connections.

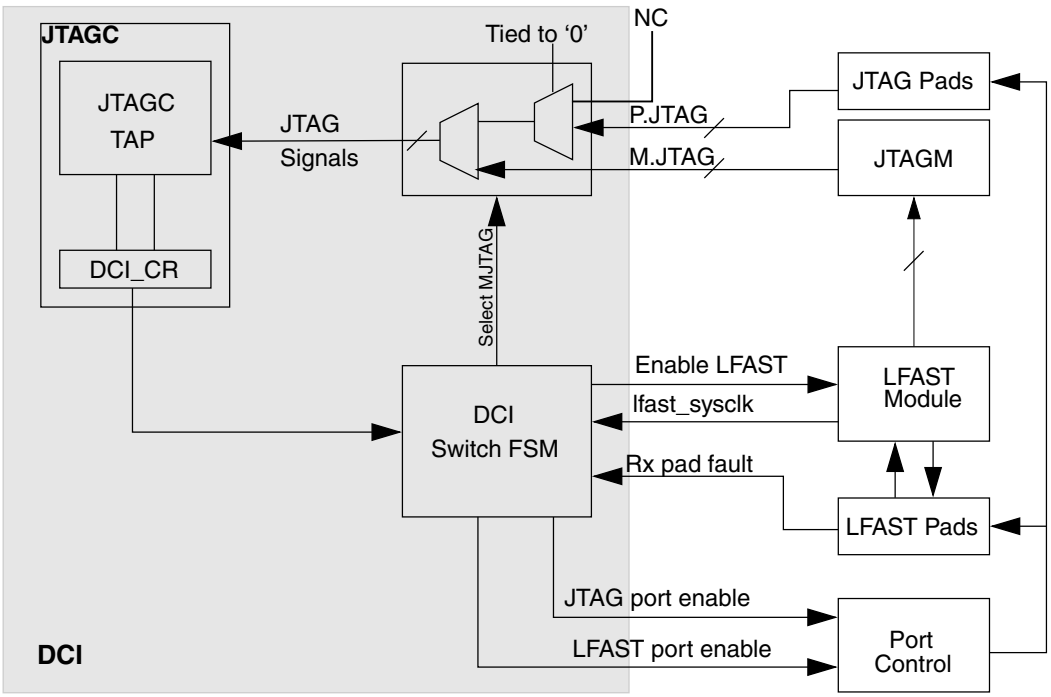


Figure 68-4. DCI JTAG/LFAST switch integration

68.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

- EN_LFAST—Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2—Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure.The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in Figure 68-5.

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

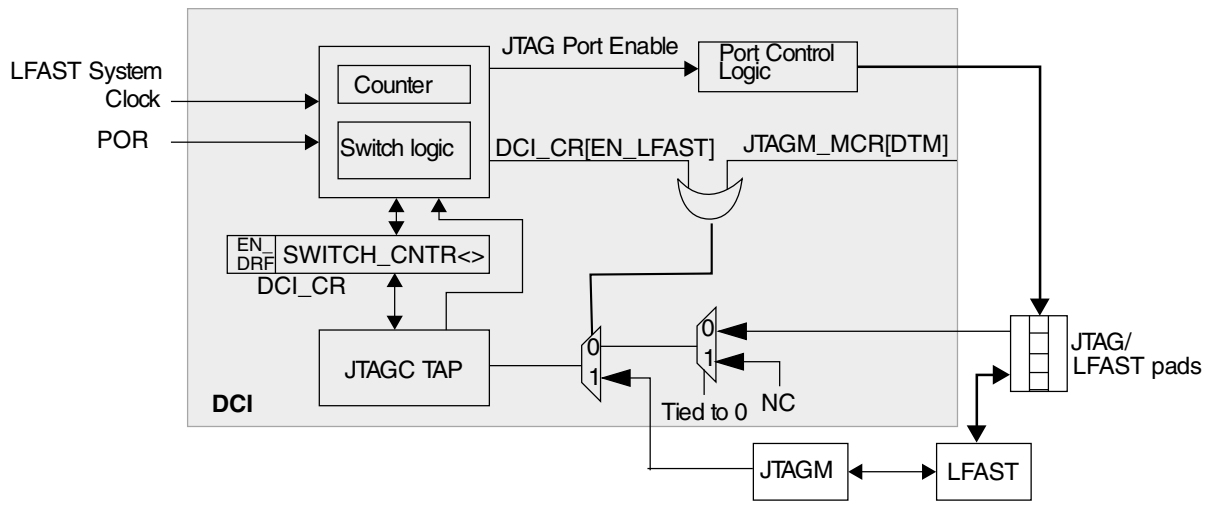


Figure 68-5. DCI switching between JTAG and LFAST modes

68.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
   - EN_LFAST
   - SWITCH_CNTR1
   - SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.

5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

This figure shows the timing sequence.

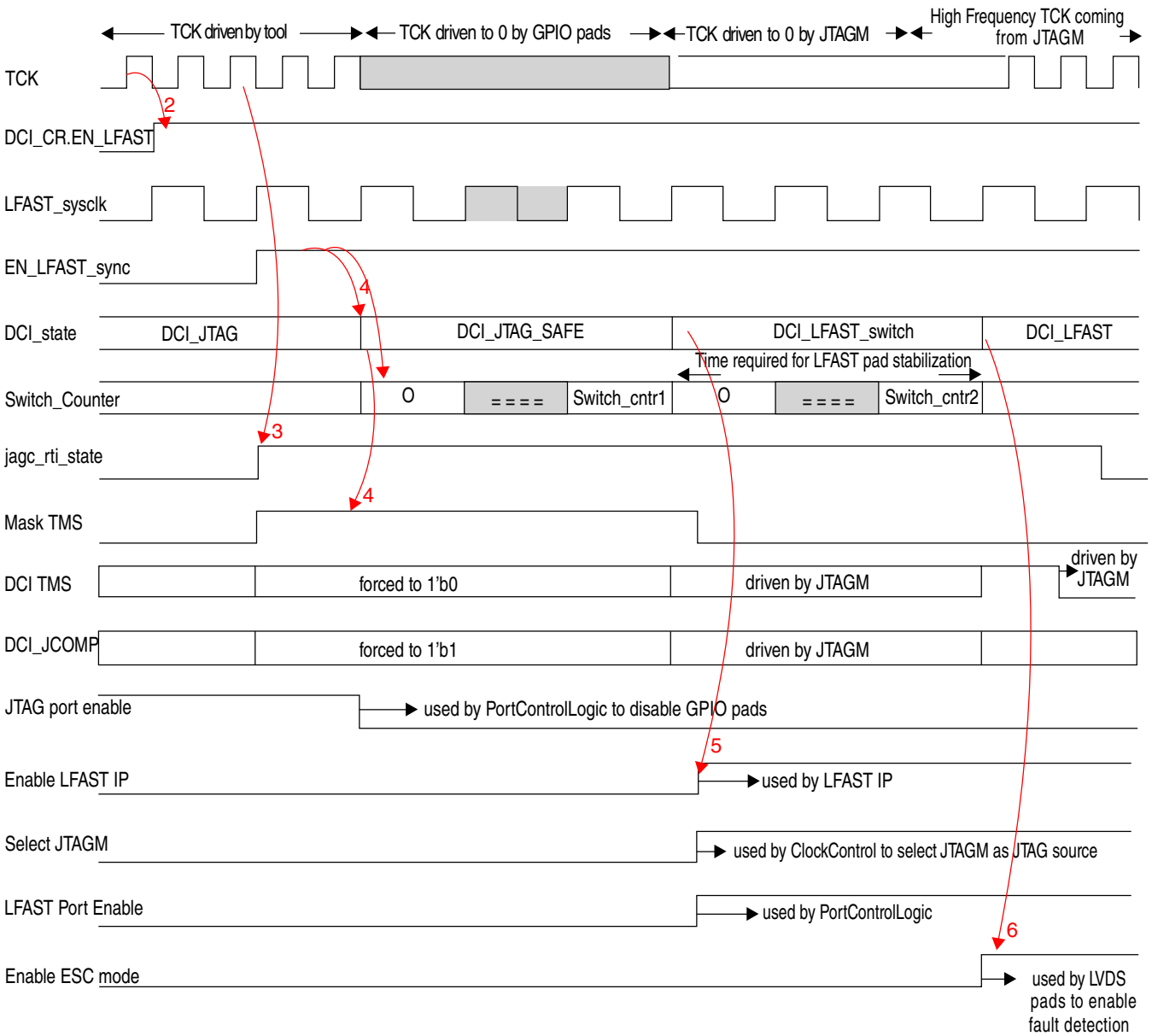


Figure 68-6. DCI JTAG to LFAST switching diagram

68.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.

3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

Table 68-7 and Table 68-9 summarize the MCU and tool behaviors. The following figure shows the timing sequence.

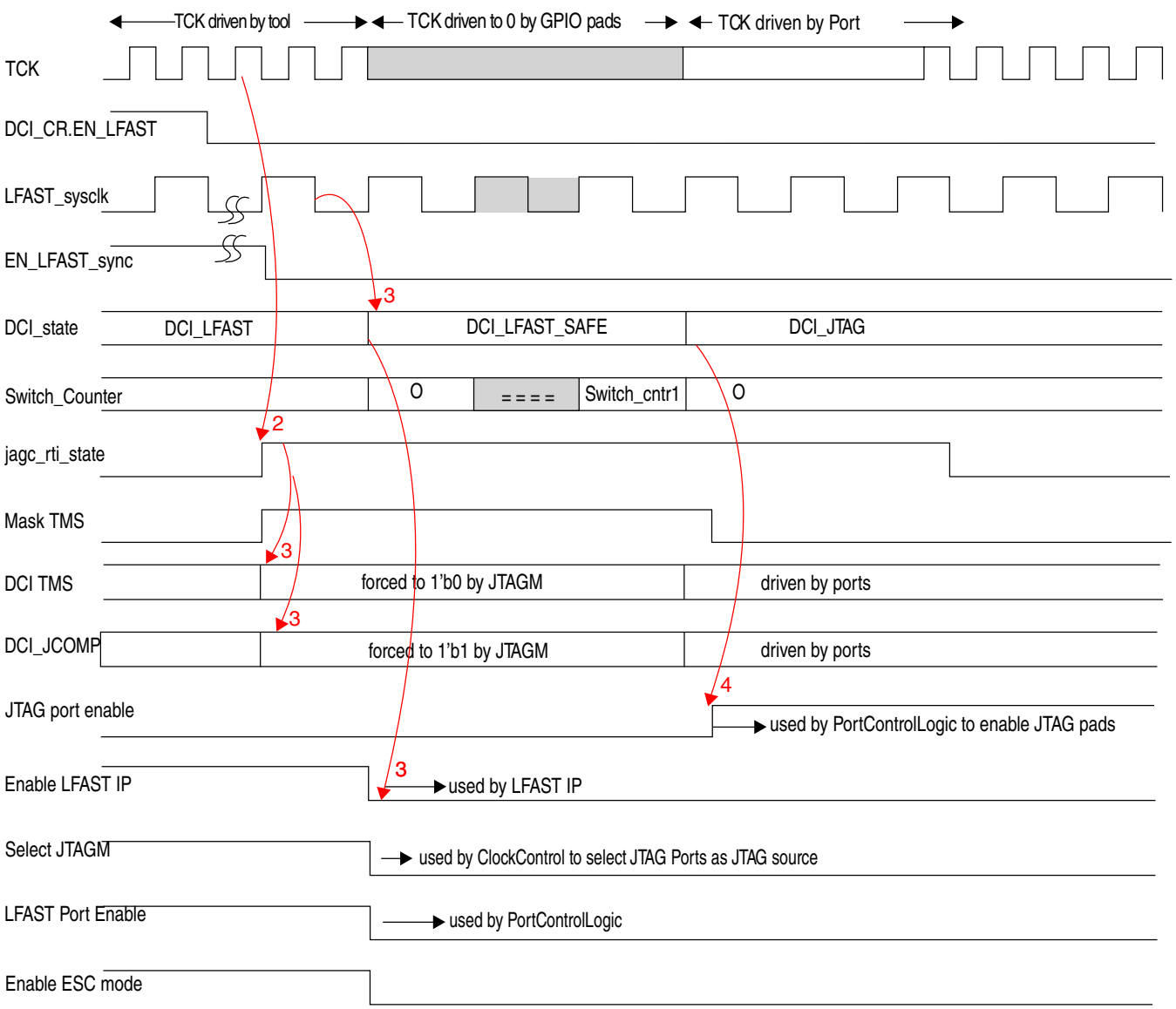


Figure 68-7. DCI LFAST to JTAG switching diagram

68.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

68.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

NOTE

It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

68.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 68-4. Nexus Reset Truth Table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

68.1.3.5 System watchdog control

The DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the $\overline{\text{EVTI}}$[0] pin is used. If the $\overline{\text{EVTI}}$[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the $\overline{\text{EVTI}}$[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the $\overline{\text{EVTI}}$[0] pin.

68.1.3.6 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as timers, DMA, Watchdog and other peripherals.

There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[1:0] signals from EVTI pads
- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor cores
- All break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

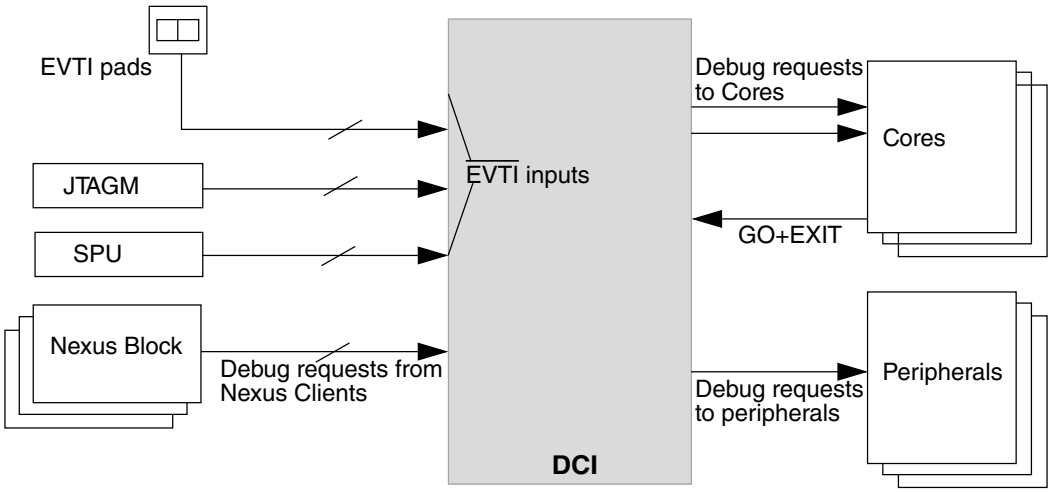


Figure 68-8. DCI debug control mechanism

The various sources of $\overline{\text{EVTI}}$[1:0] inputs are synchronized and logically ANDed in the DCI to generate two debug requests $\overline{\text{DCI\_EVTI}}$[1:0] for the processor cores and non-CPU peripherals. Of these two inputs, $\overline{\text{EVTI}}$[0] is used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The other input, $\overline{\text{EVTI}}$[1], is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted. The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral debug request is negated when the processor core(s) execute GO+EXIT command. The $\overline{\text{EVTI}}$[1:0] signals coming from the pads are enabled for debug control using DCI_CR[5:4] bits.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

This figure shows the timing sequence.

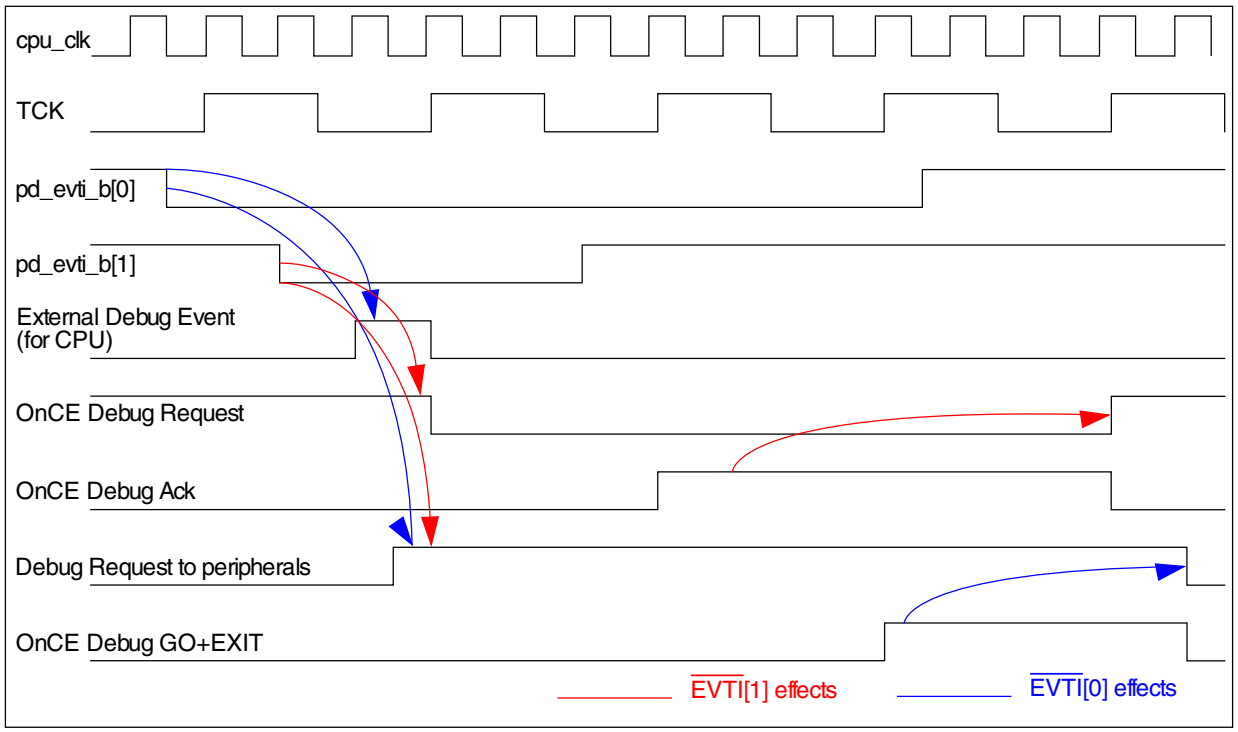


Figure 68-9. DCI debug signal generation

68.1.3.7 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

68.1.3.8 Synchronous restart control

The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (i.e. set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

68.1.3.9 EVTO management

The EVTO management block diagram is presented in the following figure.

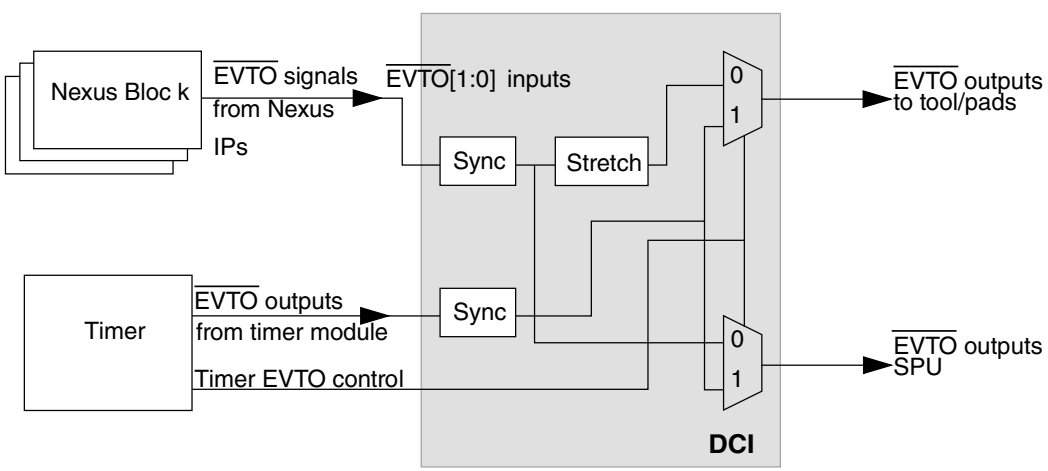


Figure 68-10. DCI EVTO management

The DCI module controls sharing of the EVTO outputs between all Nexus Clients including the cores that produce $\overline{\text{EVTO}}$[1:0] signals. The DCI uses a high frequency clock to sample EVTO signals coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted $\overline{\text{EVTO}}$[1:0] signals from any Nexus client, the DCI immediately asserts $\overline{\text{EVTO}}$[1:0] outputs and stretches it for a minimum of one output clock period for the output to the

tool(Figure 68-11). The DCI also generates EVTO outputs in other device clock domains so that other Nexus clients can receive EVTO inputs without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

The DCI also allows the timer to take control of the EVTO outputs by controlling a timer register. If these fields are programmed to HALT/TIM/TOM/ATOM EVTO value, the timer EVTO outputs are directly available on the ports bypassing the stretch logic.

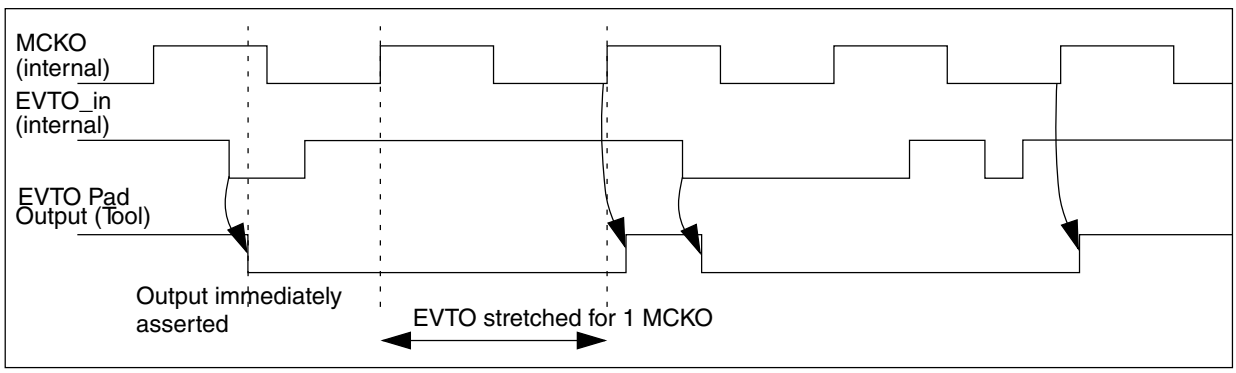


Figure 68-11. DCI EVTO timing diagram

68.1.3.10 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.
- Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

68.1.3.11 Security

The SoC security logic provides a Debug_Enable signal to the DCI. The DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

68.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

68.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

68.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. The following table shows the format of the DCI_CR.

	31	30	29	28	27	26	25	24
R W	SWITCH_CNTR2							
Reset	1	1	1	1	1	1	1	1
	23	22	21	20	19	18	17	16
R W	SWITCH_CNTR2							
Reset	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8
R W	SWITCH_CNTR1					FLVDS_PAD_FLT	FPD_FUNC	FPD_DEST
Reset	1	1	1	1	1	0	0	0
	7	6	5	4	3	2	1	0
R W	Reserved	FORCE_LFAST	EVTI1_PAD_EN	EVTI0_PAD_EN	EVTO1_PAD_EN	EVTO0_PAD_EN	EN_ESC_MODE	EN_LFAST
Reset	0	0	0	0	0	0	0	0

The DCI_CR is described in this table.

Table 68-5. DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2. Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = 32,768×T_{lfast_sysclk}
15:11 SWITCH_CNTR1	Width of Switch_CNTR1. Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = 32×T_{lfast_sysclk}
10 FLVDS_PAD_FLT	Force LVDS Pad Fault. Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
9 FPD_FUNC	Force functional reset 0 Do not force functional reset 1 Force functional reset
8 FPD_DEST	Force destructive reset 0 Do not force destructive reset 1 Force destructive reset

Table continues on the next page...

Table 68-5. DCI_CR field descriptions (continued)

Field	Description
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
5 EVTI1_PAD_EN	$\overline{\text{EVTI}}$[1] feature enable 0 $\overline{\text{EVTI}}$[1] disabled (default) 1 $\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN[1]	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled (default) 1 $\overline{\text{EVTI}}$[0] enabled
3 EVTO1_PAD_EN	Output enable for $\overline{\text{EVTO}}$[1] pad 0 $\overline{\text{EVTO}}$[1] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[1] pad output enabled
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}$[0] pad 0 $\overline{\text{EVTO}}$[0] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[0] pad output enabled
1 EN_ESC_MODE	Enable Escape Mode. Fault detect feature enable for LFAST Rx pads 0 Fault detect feature not present for LFAST Rx pads (default) 1 Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST[2]	Enable for LFAST-based tool selection. This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset. 0 LFAST tool not enabled. DCI operates in JTAG mode (default) 1 LFAST tool enabled

1. In an ED that has both PD and BD, EVTI1_PAD_EN and EVTI0_PAD_EN must be programmed in both BD and PD instances of the DCI in order to allow events generated on the BD to pass through the BD DCI and reach the PD DCI.
2. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

68.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

68.4 Functional description

This section describes the functional description of the DCI.

68.4.1 DCI mode transition

After power-on reset, the DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. Table 68-6 and Table 68-7 show the steps and expected states for the MCU and tool for JTAG to LFAST switching. Table 68-8 and Table 68-9 contains the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 68-6. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_ SYSCLK _EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 68-7. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode)[1]	DCI_CR[EN_ LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 68-8. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input

Table continues on the next page...

Table 68-8. MCU operation (LFAST to JTAG switching) (continued)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 68-9. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

68.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode.Table 68-10 and Table 68-11 contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 68-10. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input

Table continues on the next page...

Table 68-10. MCU operation (LFAST to JTAGRST switching) (continued)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
Mode Escape	Enable	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 68-11. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don't Care
TMS	LVDS TxP	Don't Care
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don't Care

Chapter 69
JTAG Controller (JTAGC)

69.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

69.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

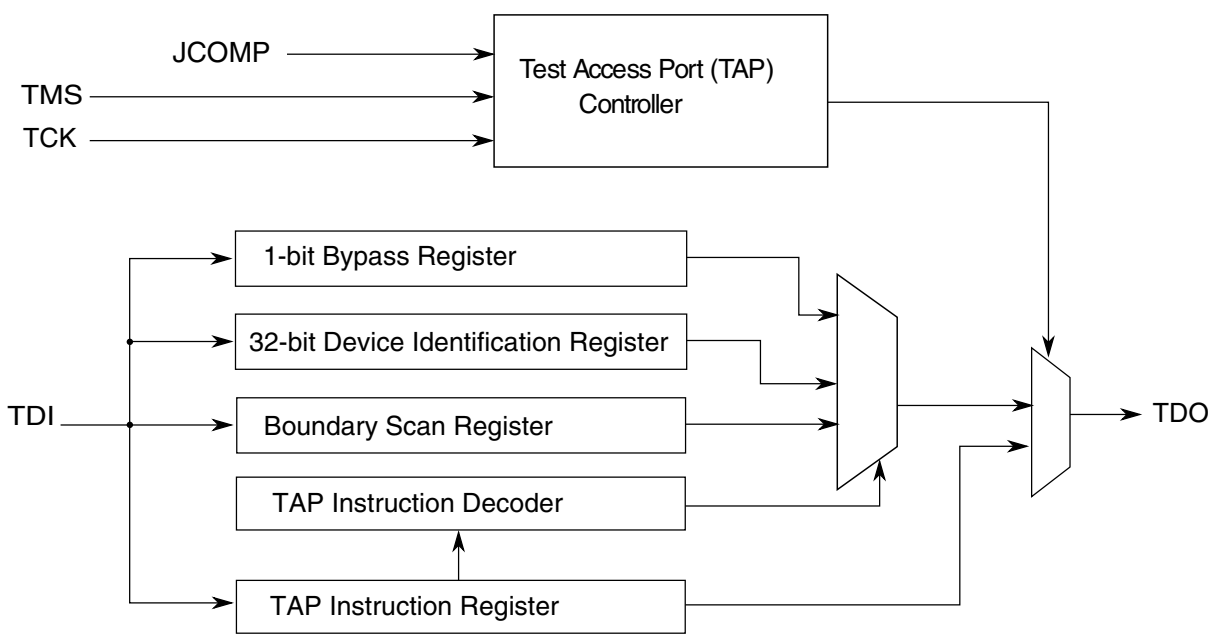


Figure 69-1. JTAG (IEEE 1149.1) block diagram

69.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 69-4 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Bypass register, boundary scan register, data registers, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

69.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

69.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

69.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

69.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

69.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 69-1. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down

Table continues on the next page...

Table 69-1. JTAG signal properties (continued)

Name	I/O	Function	Reset State	Pull
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z[1]	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

69.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

69.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

69.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

69.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

69.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

69.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

69.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

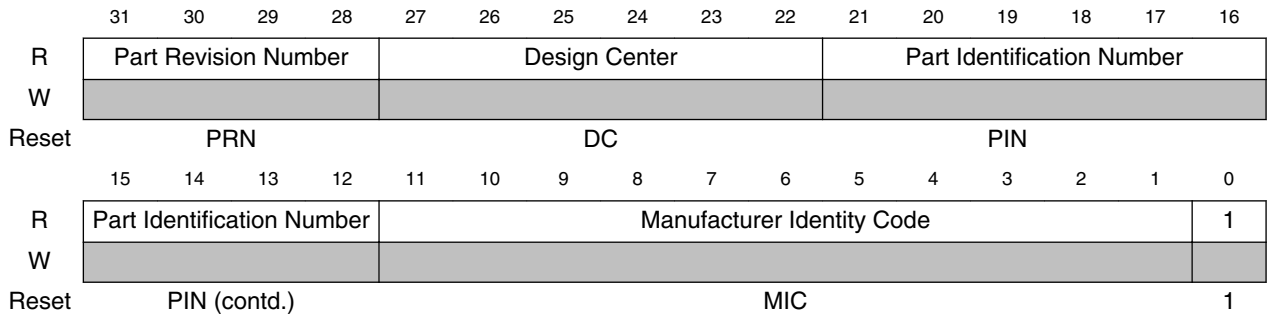
Figure 69-2. Instruction register

69.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

69.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



The following table describes the device identification register functions.

Table 69-2. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x2B.
PIN	Part Identification Number. Contains the part number of the device. 0x30E.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

69.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.

	255	...	2	1	0
R	JTAG_PASSWORD				
W					
Reset:	*[1]	*	*	*	*

1. The reset value of JTAG_PASSWORD is 256 'b0.
1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 69-3. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

69.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

69.4 Functional description

This section explains the JTAGC functional description.

69.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

69.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

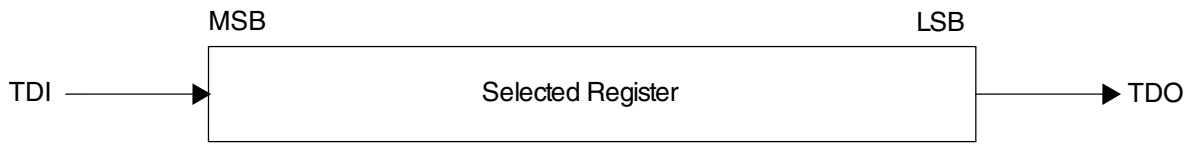
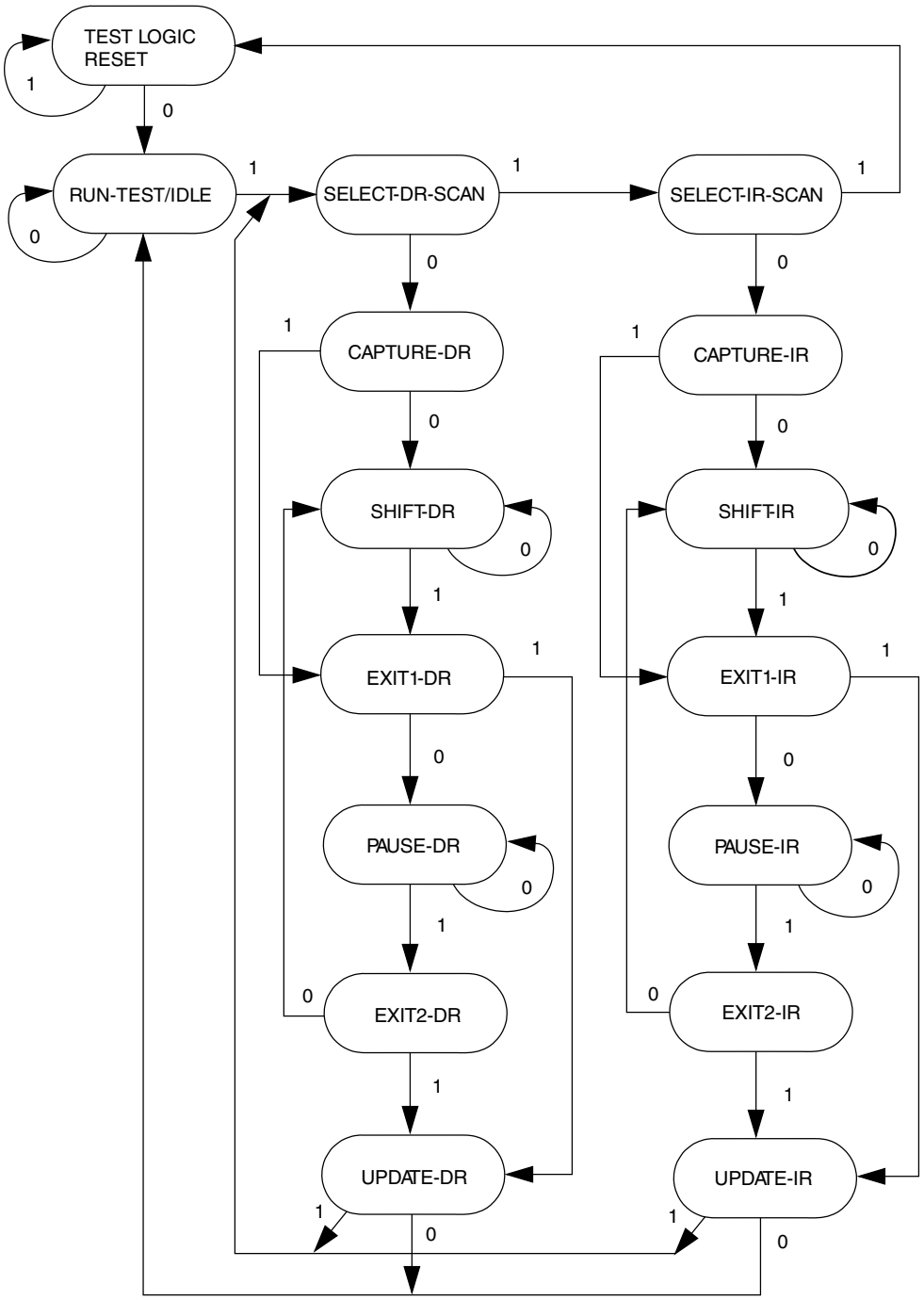


Figure 69-3. Shifting data through a register

69.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 69-4. IEEE 1149.1-2001 TAP controller finite state machine

69.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

69.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

69.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 69-4. General 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	001010	Intended for factory debug only
CLAMP	001100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.

Table continues on the next page...

Table 69-4. General 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ENABLE_DCI_PINCR	001111	Enables access to DCI PINCR register
BYPASS	111111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

69.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

69.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

69.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

69.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

69.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

69.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

69.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a

single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

69.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

69.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

69.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

69.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller.
2. Load the appropriate instruction for the test or action to be performed.

Chapter 70
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

70.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

70.2 Abbreviations

Table 70-1. Abbreviations

Abbreviation	Expansion
APU	Advanced Protocol Unit
APFC	Auxiliary Pin Function Control
CGM	Conditional Group Member
CGMC	Conditional Group Membership Count
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-Level TAPC
CP	Check Packet
CPA	Check Process Active
DCU	Data Channel Unit
EOT	End of Transfer
EPU	Extended Protocol Unit
Jscan	JTAG Scan
Mscan	Maximum-flexibility Scan
nTRST	Active low Test Reset
Oscan	Optimized Scan
PSS	Pause Selection State

Table continues on the next page...

Table 70-1. Abbreviations (continued)

Abbreviation	Expansion
RSU	Reset and Selection Unit
SP	Scan Packet
SPA	Scan Packet Active
Sscan	Segmented Scan
SSD	Scan Selection Directive
SSM	Scan State Machine
STL	System Test Logic
TAP	Test Access Port
TAP.1	An IEEE 1149.1 Test Access Port
TAP.7	An IEEE 1149.7 Test Access Port
TCA	Tap Controller Address
TCK	Test Clock
TCKC	Test Clock Compact
TDI	Test Data Input
TDIC	Test Data Input Compact
TDO	Test Data Output
TDOC	Test Data Output Compact
TMS	Test Mode Select
TMSC	Test Mode Select Compact
TS	Target System
ZBS	Zero Bit Scan

70.3 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as "Compact JTAG" or "CJTAG".

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths

- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

70.3.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- **Compliant**
  - Compliant with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- **Extended**
  - Compatible with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  - Additional test/debug functions
- **Advanced**
  - Two-pin operation using the "Advanced Protocol" — signaling where TCK(C) and TMS(C) signals transfer control/data information
  - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

70.3.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors..
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

70.3.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in the following table. Signal names that end in “C” are associated with expanded functionality (above that provided by the IEEE 1149.1 standard).

The notation used in the table is as follows:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the TCKC, TMSC, TDIC, TDOC and nTRST pins.

Table 70-2. TAP.7 signal list

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M

Table continues on the next page...

Table 70-2. TAP.7 signal list (continued)

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

70.3.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in the following figure. It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP. 7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

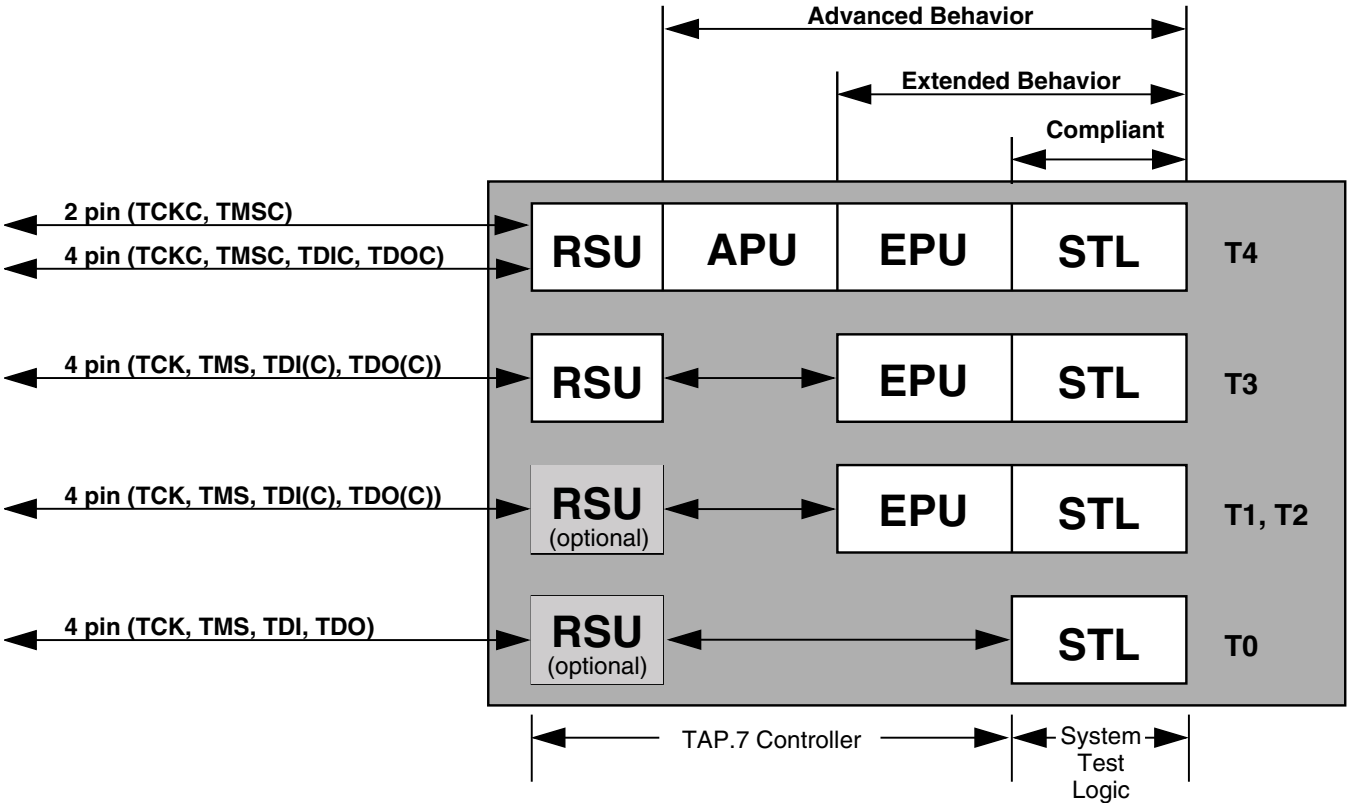


Figure 70-1. TAP.7 Controller Architecture

70.3.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in the following figure.

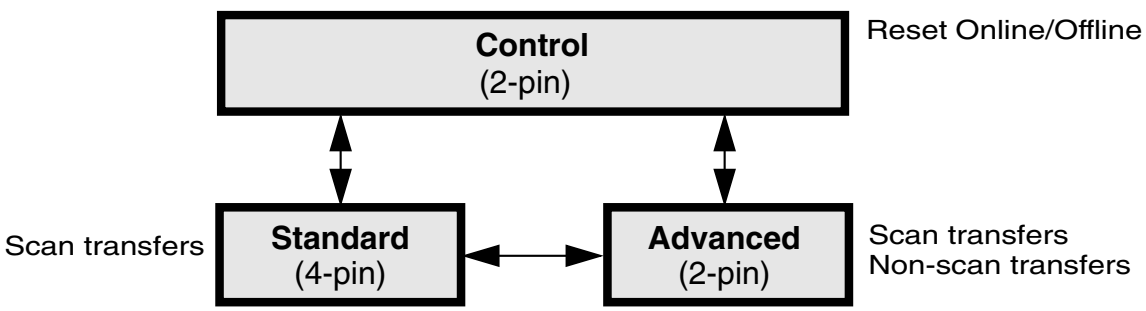


Figure 70-2. TAP.7 Behaviors

The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and

vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in the following table.

Table 70-3. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

70.3.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

70.3.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

70.3.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

70.4 Operating models

The following figure illustrates a simplified view of the CJTAG operating model.

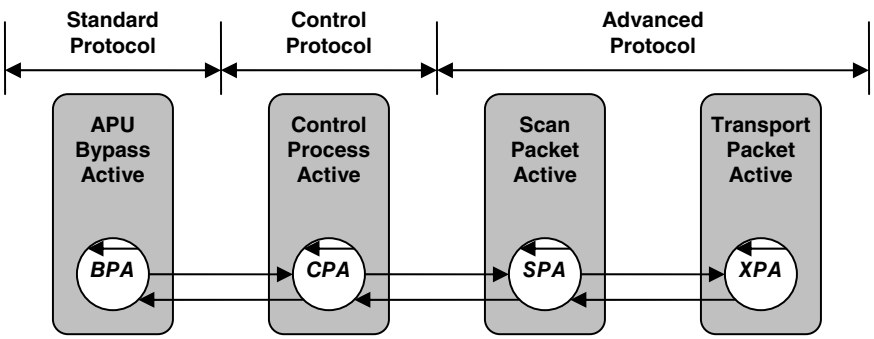


Figure 70-3. CJTAG operating model

70.5 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

70.5.1 T0 functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

70.5.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

70.5.3 T2 functions

- "Super Bypass" function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

70.5.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

70.5.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

70.6 Ancillary services

70.6.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options

- Escape sequences
- TAPC state machine

70.6.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC.

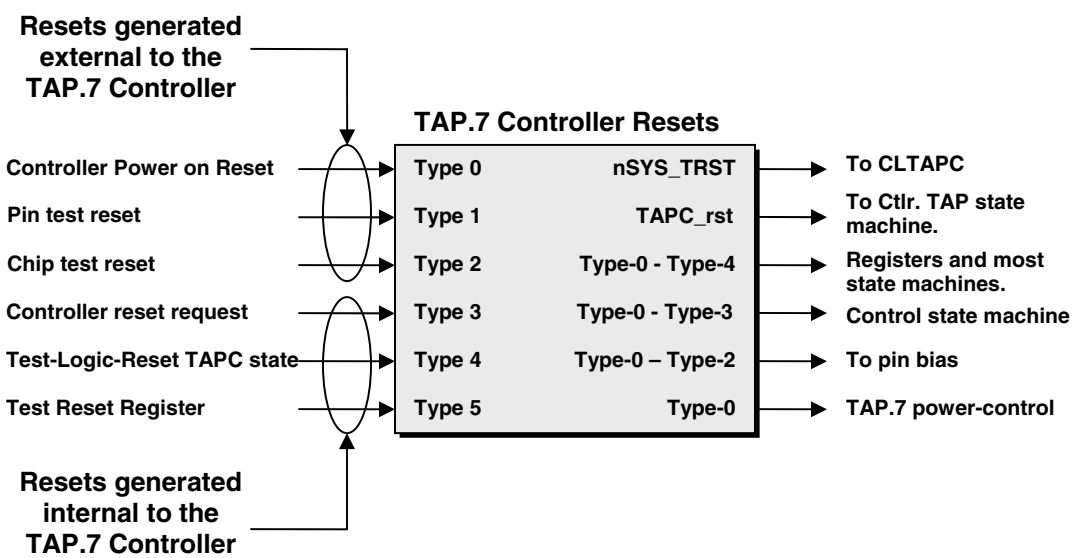


Figure 70-4. Conceptual view of the EPU reset logic and state machines

70.6.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

70.6.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

70.6.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

70.6.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

70.6.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

70.6.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

70.6.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

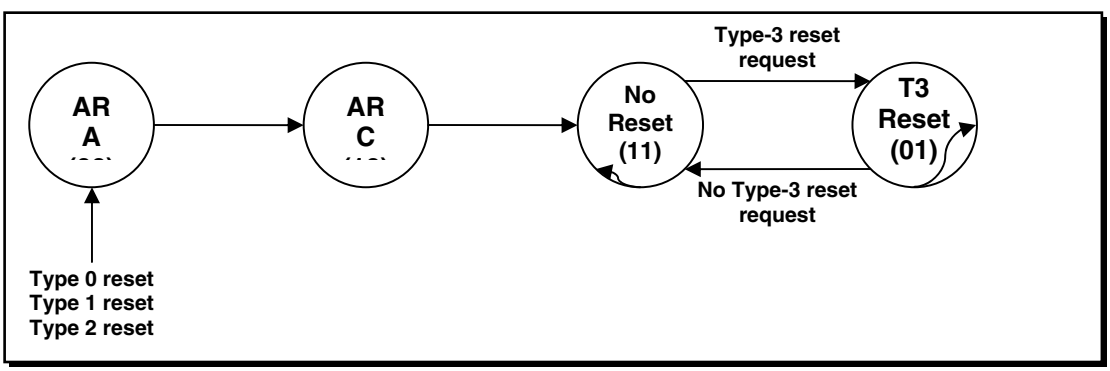


Figure 70-5. Reset state machine

Table 70-4. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in the following table.

Table 70-5. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

70.6.3 Start-up Options

70.6.3.1 Overview

The TAP.7 start-up options are shown in the following table. The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 70-6. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled

Table continues on the next page...

Table 70-6. Start-up Options (continued)

Start-up Option	Description
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

70.6.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 70-7. Reset values for TAP.7 startup options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

70.6.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

70.6.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP. 7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in the following figure.

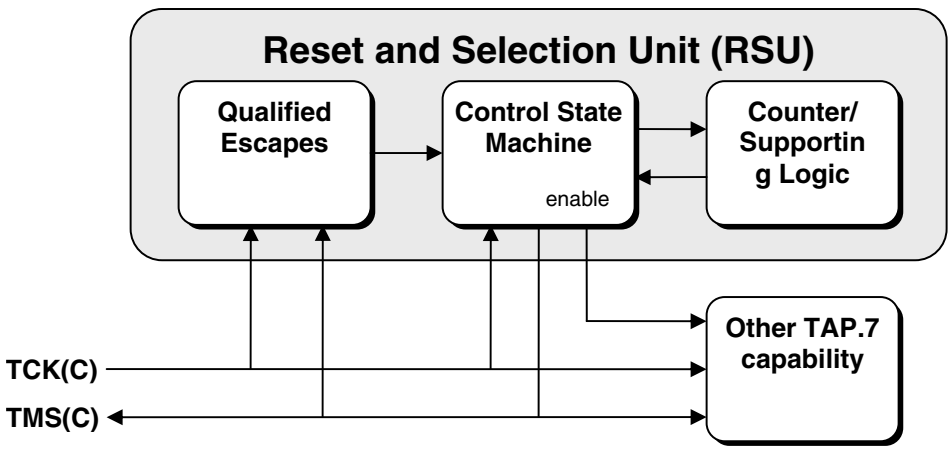


Figure 70-6. Conceptual view of technology selection mechanism

70.6.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

70.6.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

70.6.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

70.6.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

70.6.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

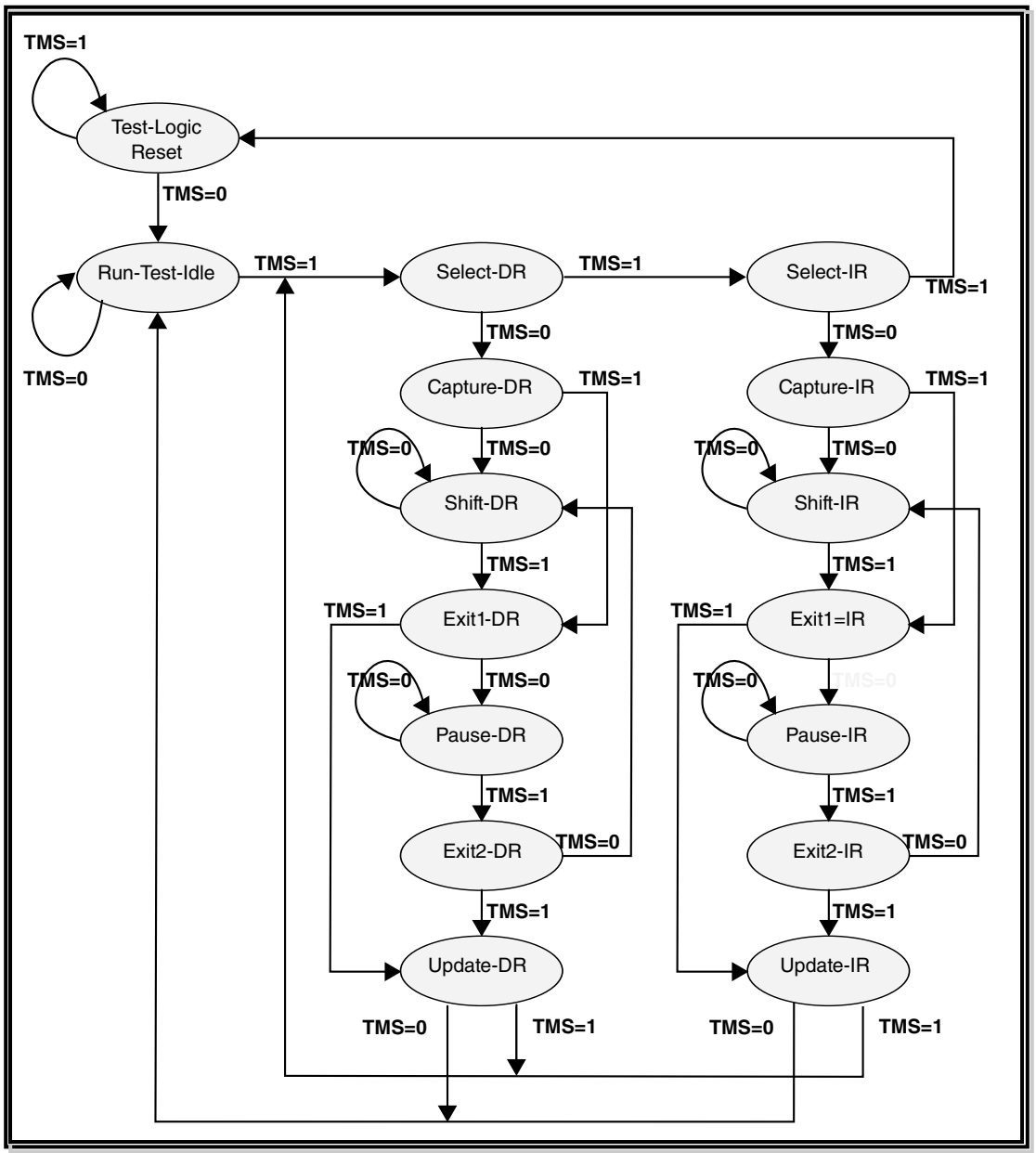


Figure 70-7. Falling edge TAP controller state machine

70.7 EPU (Extended Protocol Unit) Operation

70.7.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

70.7.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

70.7.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

70.7.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

70.7.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths

6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control levels.

70.7.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

70.7.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

70.7.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "***TAP.7 ZBSs***". ZBSs that are intended for other uses are henceforth called "***Other ZBSs***". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP. 7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

70.7.2 EPU Registers

70.7.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

70.7.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back..

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 70-8. TAP.7 controller register list (managed with commands)

Register type	Global/local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	RDBACK1	Read-back 1
		32	CNFG0	Configuration Register 0

M = Mandatory register

O = Optional register

70.7.2.3 Register Descriptions

70.7.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

70.7.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the "System Path" (STL) is used.

1 – Suspend use of control levels. Enables "System Path."

0 – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

70.7.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

1 – Inhibit ZBS detection.

0 – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

70.7.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 70-9. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	**JScan0:** • 1149.1 Compliant Operation. • Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	**JScan1:** • 1149.1 Compatible Operation • Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	**JScan2:** • 1149.1 Compatible Operation. • Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	**JScan3:**

Table continues on the next page...

Table 70-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
		• Non-compatible 1149.1 Operation • Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. • Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. • Enables the use of SSDs
T4	01000	**OScan0** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01001	**OScan1** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01010	**OScan2** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications
	01011	**OScan3** • TAP.1 components • No TDI in non-shift states • No TDO in any state
T4	01100	**OScan4** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01101	**OScan5** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01110	**OScan6** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications

Table continues on the next page...

Table 70-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
	01111	**OScan7** • TAP.1 components • No TDI in non-shift states • No TDO in any state
	10000	**MScan** • Data rate dependent components • Multi TAP.7 communication • Directed CID assignment • Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	**Reserved**

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

70.7.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

0 – Scan selection directives are not enabled.

1 – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

70.7.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the "Global Register."

70.7.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

70.7.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

0 – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.

1 – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

70.7.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

0x – TDI and TDO have JTAG function.

1x – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

70.7.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

70.7.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

70.7.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

70.7.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as "0".

Table 70-10. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

70.7.2.3.14 Configuration Register 0 (CNFG0)

Table 70-11. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class 0000 – TAP.1 or TAP.7 Class T0 0001 – Class T1 0010 – Class T2 0011 – Class T3 0100 – Class T4

Table continues on the next page...

Table 70-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision.
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes

Table continues on the next page...

Table 70-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported

Table continues on the next page...

Table 70-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
Reserved	31:29	3	Reserved	Reserved, read as zero

70.7.2.3.15 Register reset values

Table 70-12. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DCA	DCA	DCA	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

70.7.3 EPU Commands

There are two command types:

- Two Part Commands
- Three Part Commands

Any number of two-part and three part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

70.7.3.1 Two Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

70.7.3.2 Three Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

70.7.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of Shift-DR states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in the following figure.

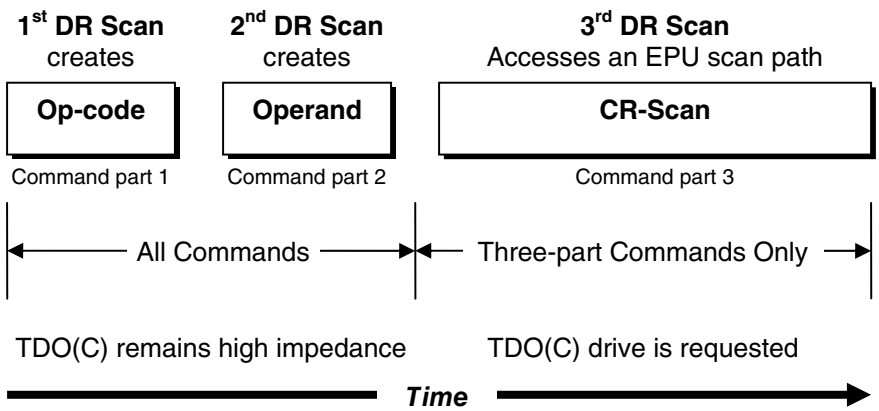


Figure 70-8. DR_Scan sequence used for command creation

The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two part commands, but only three three-part commands, each having a special purpose.

70.7.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in the following table.

Table 70-13. – TAP.7 controller command list

Command class	Opcode count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate[1]	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

70.7.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing an external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

70.7.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

70.7.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

70.7.3.5 Command Definitions

70.7.3.5.1 Store Commands

70.7.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 70-14. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	Operand	bbb		x	y	Description
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	-	SGC = y
				1	-	CGM = y

Table continues on the next page...

Table 70-14. Store Miscellaneous Control (continued)

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
		010	RdyCtl	-	-	RDYC = xy
		011	DlyCtl	-	-	DLYC = xy
		100	ScnFunc	0	-	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	-	-	Reserved
		110				
		111				

70.7.3.5.1.2 Store Conditional 1-bit (STC1)

Table 70-15. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	**Operand**	**c**	**bbb**	**CAPE**	**Description**
00001	c bbb v	0	000	-	SREDGE = v
			001	-	Reserved
			010	-	TRESET = v
			011 – 111	-	Reserved
		1	–-	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

70.7.3.5.1.3 Store Conditional 2-bit (STC2)

Table 70-16. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	**Operand**	**c**	**bb**	**CAPE**	**Description**
00010	c bb vv	0	00	-	Reserved
			01	-	STCKDC = vv
			10	-	APFC = vv
			11	-	Reserved
		1	–-	0	No change
			00	1	Reserved
			01	1	STCKDC = vv

Table continues on the next page...

Table 70-16. Store Conditional 2 bit (STC2) (continued)

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
			10	1	APFC = vv
			11	1	Reserved

70.7.3.5.1.4 Store Scan Format (STFMT)

Table 70-17. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

70.7.3.5.2 Select Commands

This section describes the select commands with their operands.

70.7.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 70-18. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	Operand	m	Description
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

70.7.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 70-19. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

70.7.3.5.3 Scan Commands

70.7.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 70-20. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR) (not implemented)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

70.7.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

Table 70-21. Scan String (SCNS)

SCNS			Scan String		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only

Table continues on the next page...

Table 70-21. Scan String (SCNS) (continued)

SCNS			Scan String		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

70.7.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in the following figure:

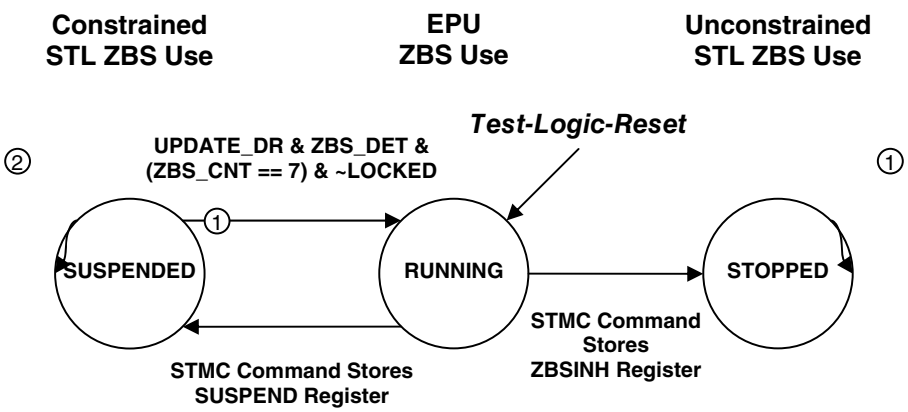


Figure 70-9. EPU operating states

The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in the following table. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 70-22. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL

Table continues on the next page...

Table 70-22. ZBS use state characteristics with both TAP.7 and other ZBS use (continued)

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

70.7.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

70.8 APU (Advanced Protocol Unit) Operation

70.8.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Two groups of scan formats (MScan and OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan Scan Format is broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a

special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

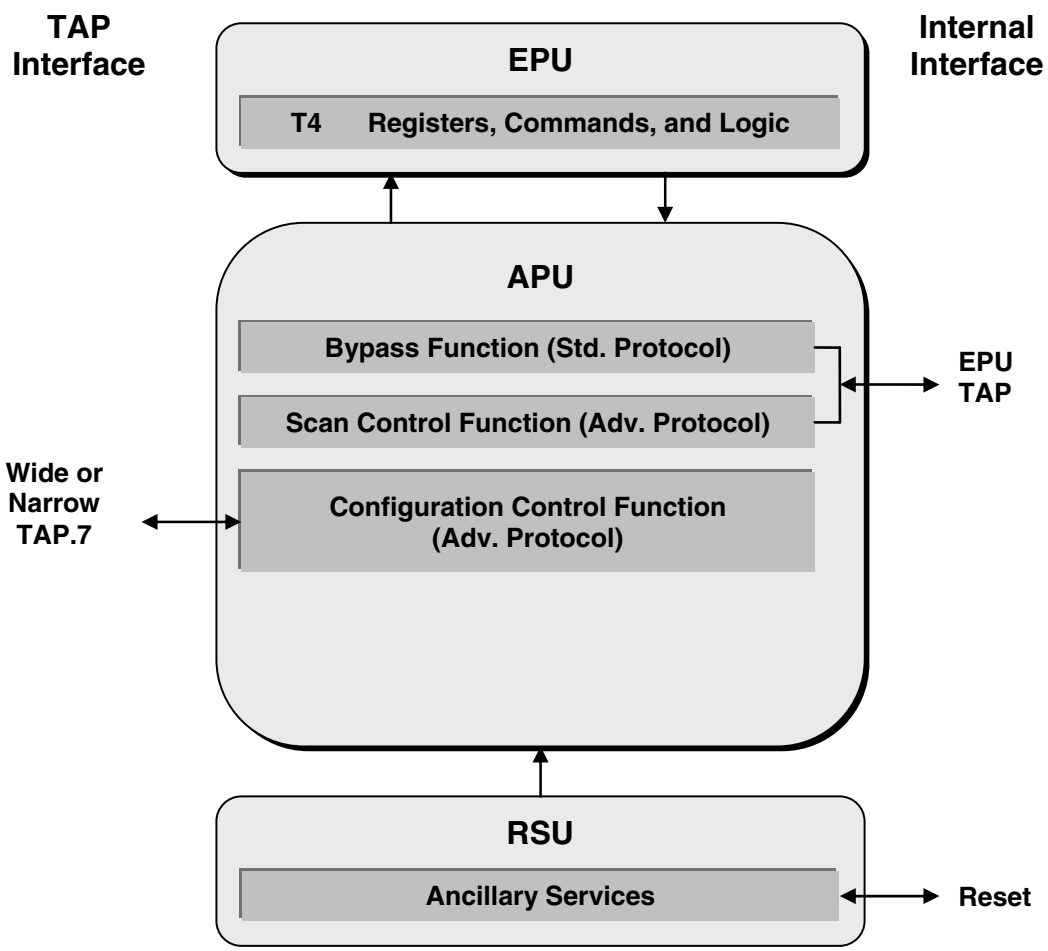


Figure 70-10. Conceptual block diagram of the APU functions

The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

70.8.2 Operation

The conceptual view of the APU operation is shown in the following figure.

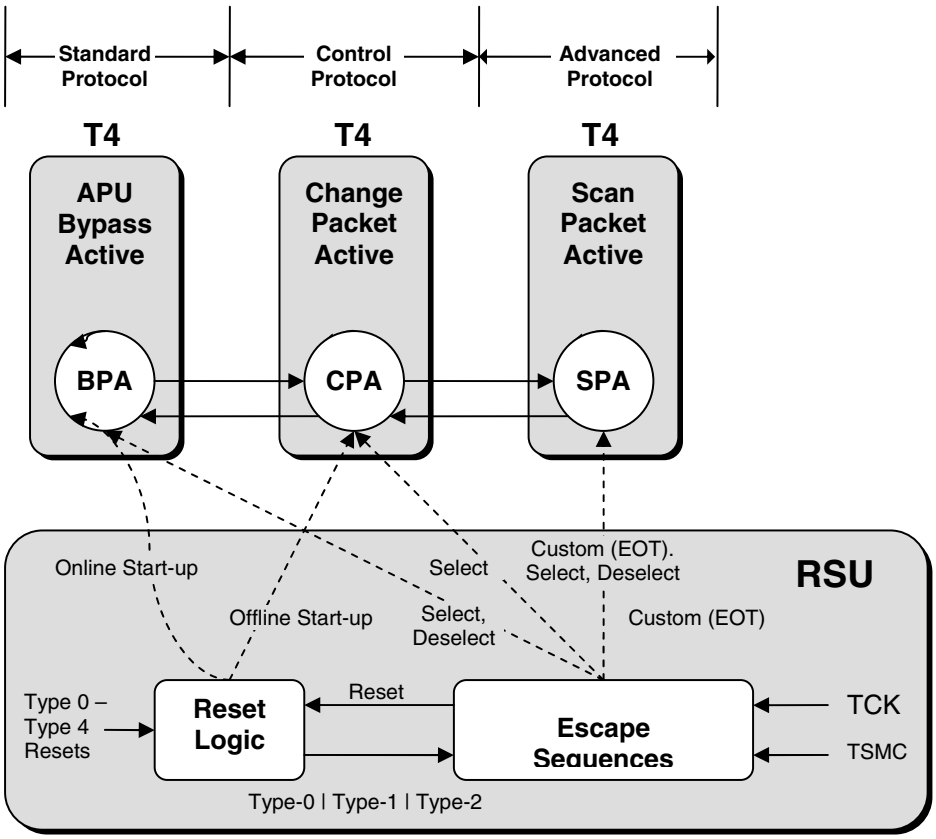


Figure 70-11. Conceptual view of APU operation

The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

70.8.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

70.8.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

70.8.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

70.8.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

70.8.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

70.8.5 APU Functions

70.8.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7 or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

70.8.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

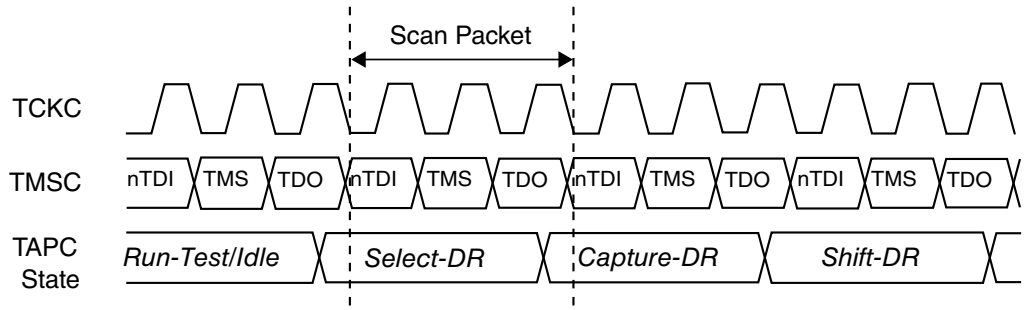


Figure 70-12. Scan Packet Sequence

A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

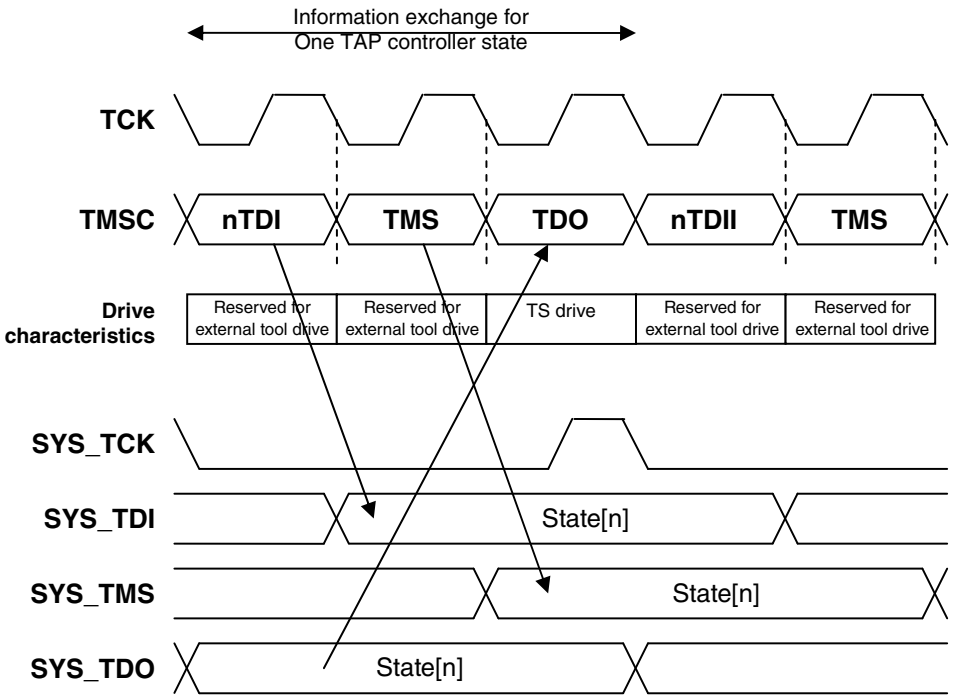


Figure 70-13. Serialization and De-serialization of Scan Packet Information

In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug

applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

70.8.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

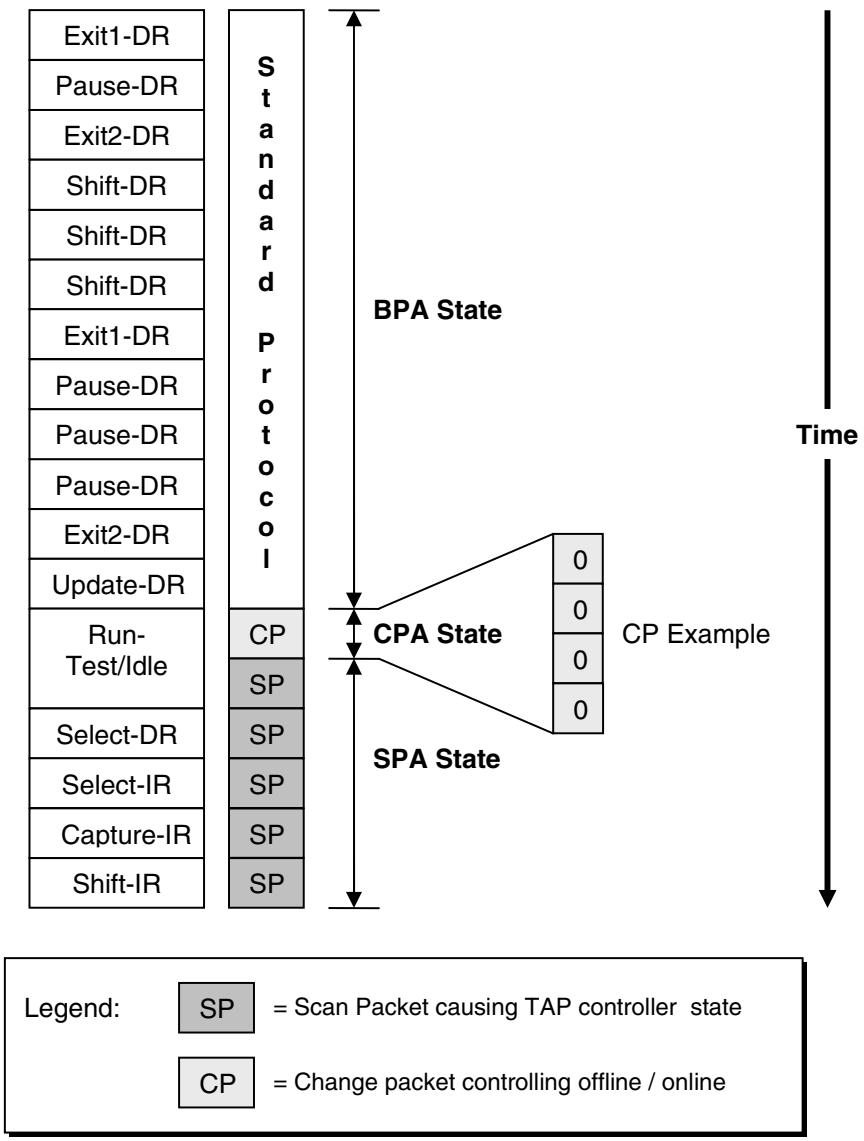


Figure 70-14. CPA State Entry from BPA State with Continued Online Operation

An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

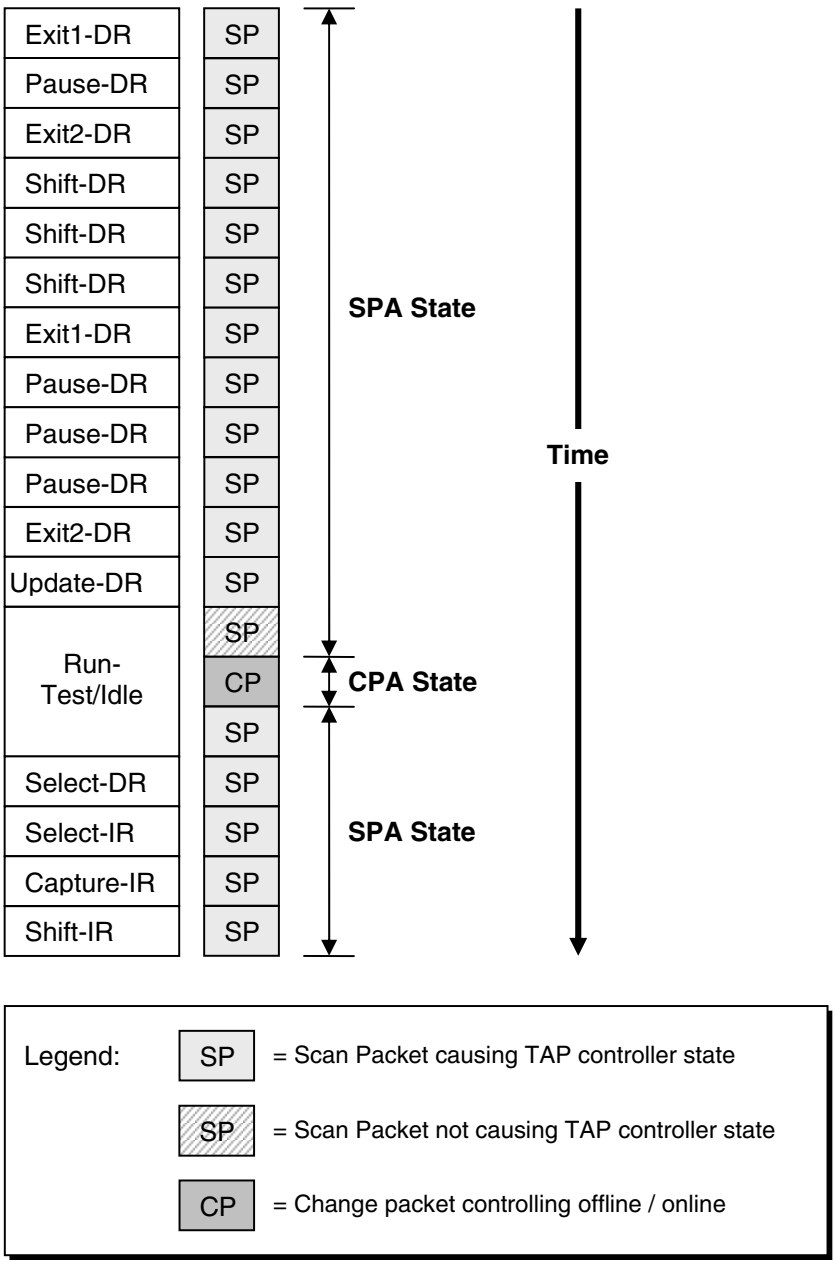


Figure 70-15. CPA State Entry from the SPA State with Continued Online Operation

When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while

the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

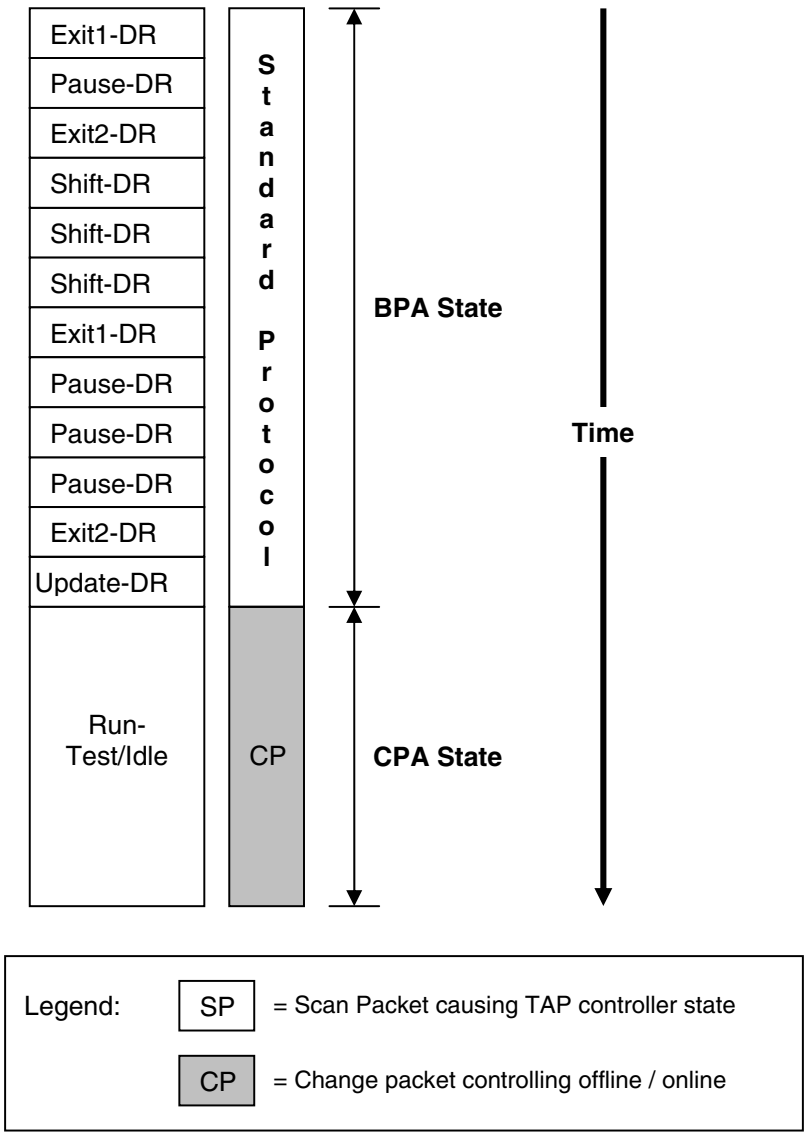


Figure 70-16. CPA State Entry from the BPA State with Offline Operation

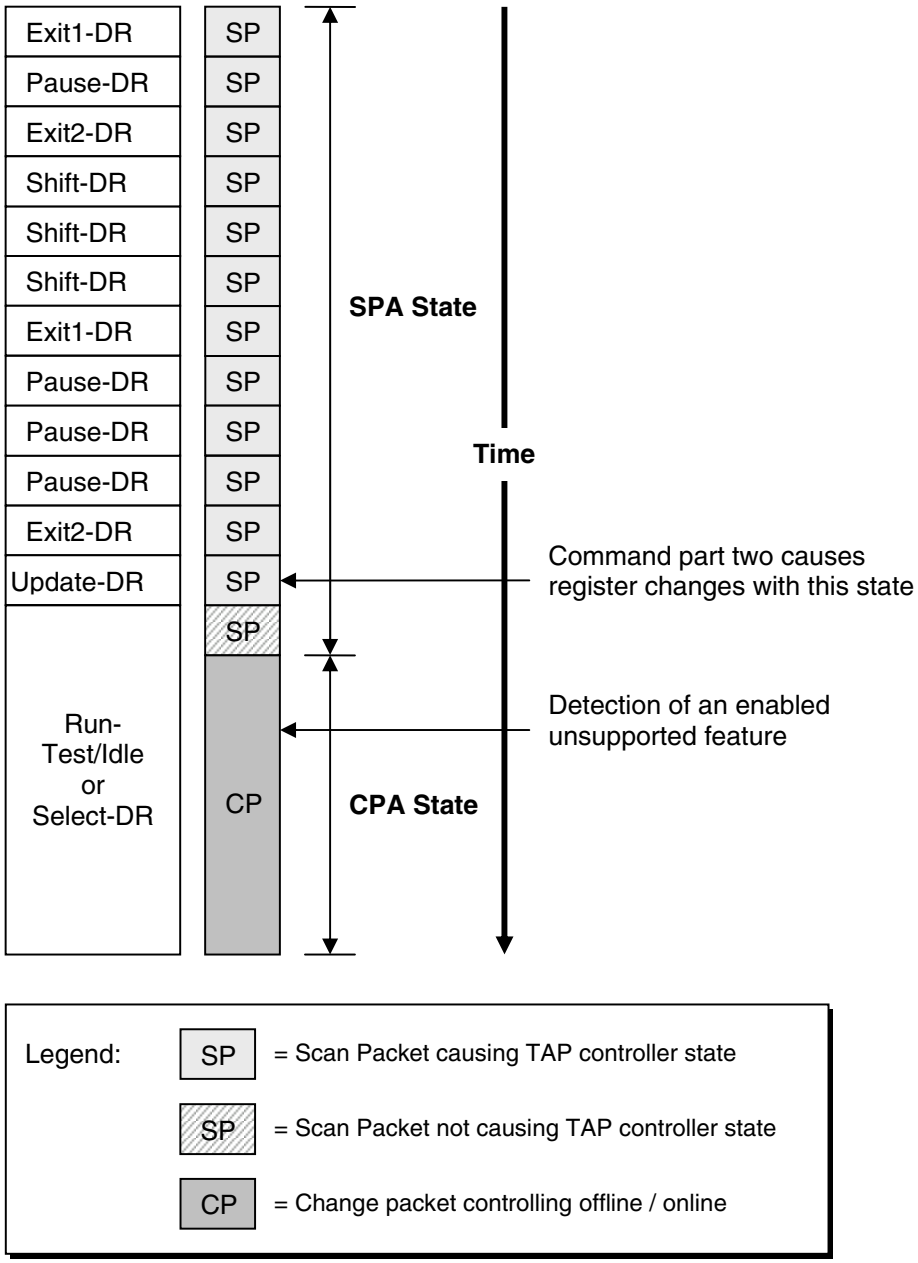


Figure 70-17. CPA State Entry from the SPA State with Offline Operation

A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

70.8.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.

- **Body** – Two or more bits representing one or more CP directives
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit.
- **Post amble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

70.8.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

70.8.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

70.8.8.1 Payload Element Content

70.8.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain and input bit frame, an output bit frame, or both. When an input bet frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

70.8.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in the following table.

Table 70-23. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

70.8.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

70.8.8.3 Scan Formats

70.8.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is

required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in Table 70-24 and Table 70-26. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

70.8.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

Table 70-24. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			

Table continues on the next page...

Table 70-24. MScan scan format use case summary (continued)

Scan format	Supporting	Performance	Flexibility	TCKC source
	Multi-chip debug communication			
	Controller ID Allocation			

Table 70-25. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

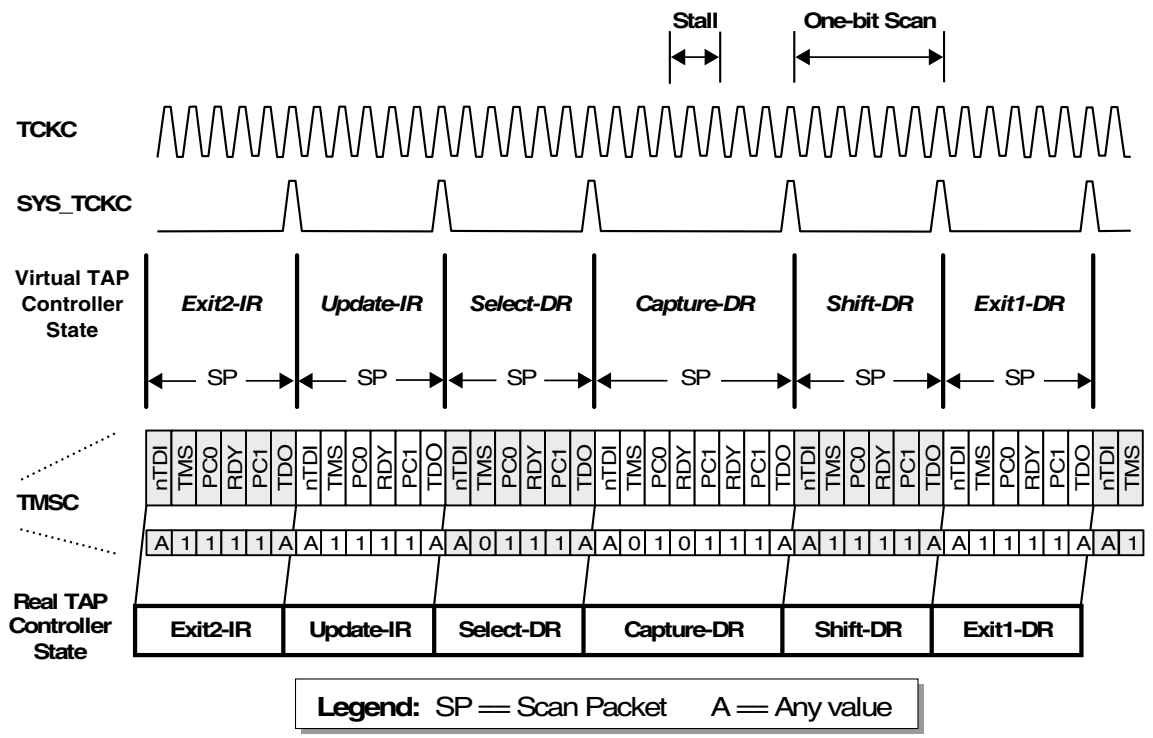


Figure 70-18. MScan Transaction

70.8.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

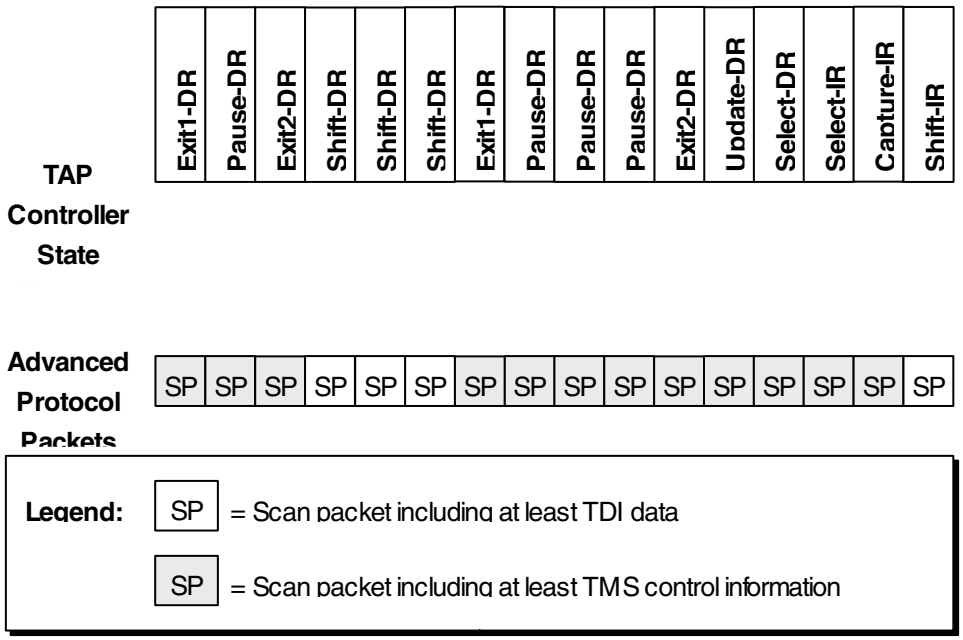


Figure 70-19. OScan Scan Packet content/TAPC state relationships

OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods fo this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in the following table.

Table 70-26. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 70-27. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	–	TDO		nTDI	TMS	–	TDO
OScan2	–	TMS	–	–		nTDI	TMS	–	TDO
OScan3	–	TMS	–	–		nTDI	TMS	–	–
OScan4	nTDI	TMS	RDY	TDO		nTDI	–	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	–	–	TDO
OScan6	–	TMS	–	–	Note 1	nTDI	TMS	–	TDO
					Note 2	nTDI	–	–	TDO
OScan7	–	TMS	–	–		nTDI	–	–	–

Note 1: Following Capture-xR or Exit2-xR Scan Packet

Note 2: Following Shift-xR Scan Packet

70.9 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

70.9.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in Control levels.

After the control level is set tocontrol level 2, TAP.7 commands can be executed as described in EPU Commands. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in Control Process Active (CPA) Function and Configuration Change Packets (CP). Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in Store Conditional 2-bit (STC2).

Chapter 71
JTAG Data Communication (JDC)

71.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

71.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JIN_IPS register, and when new data has been written to the JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in the following figure.

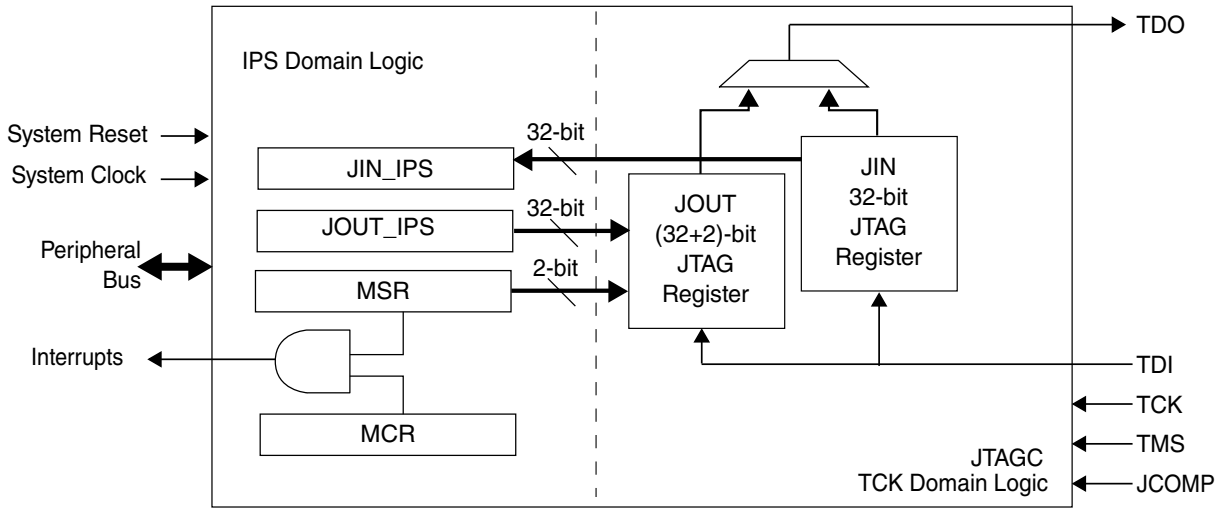


Figure 71-1. JDC block diagram

71.3 Memory mapped registers

Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

JDC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JDC_MCR)	32	R/W	0000_0000h	71.3.1/3877
4	Module Status Register (JDC_MSR)	32	R/W	0000_0000h	71.3.2/3878
8	JTAG Output Data Register (JDC_JOUT_IPS)	32	R/W	0000_0000h	71.3.3/3879
C	JTAG Input Data Register (JDC_JIN_IPS)	32	R	0000_0000h	71.3.4/3880

71.3.1 Module Configuration Register (JDC_MCR)

The MCR is used to enable the interrupt outputs of the JDC. This register is reset by system destructive reset.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															JIN_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															JOUT_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

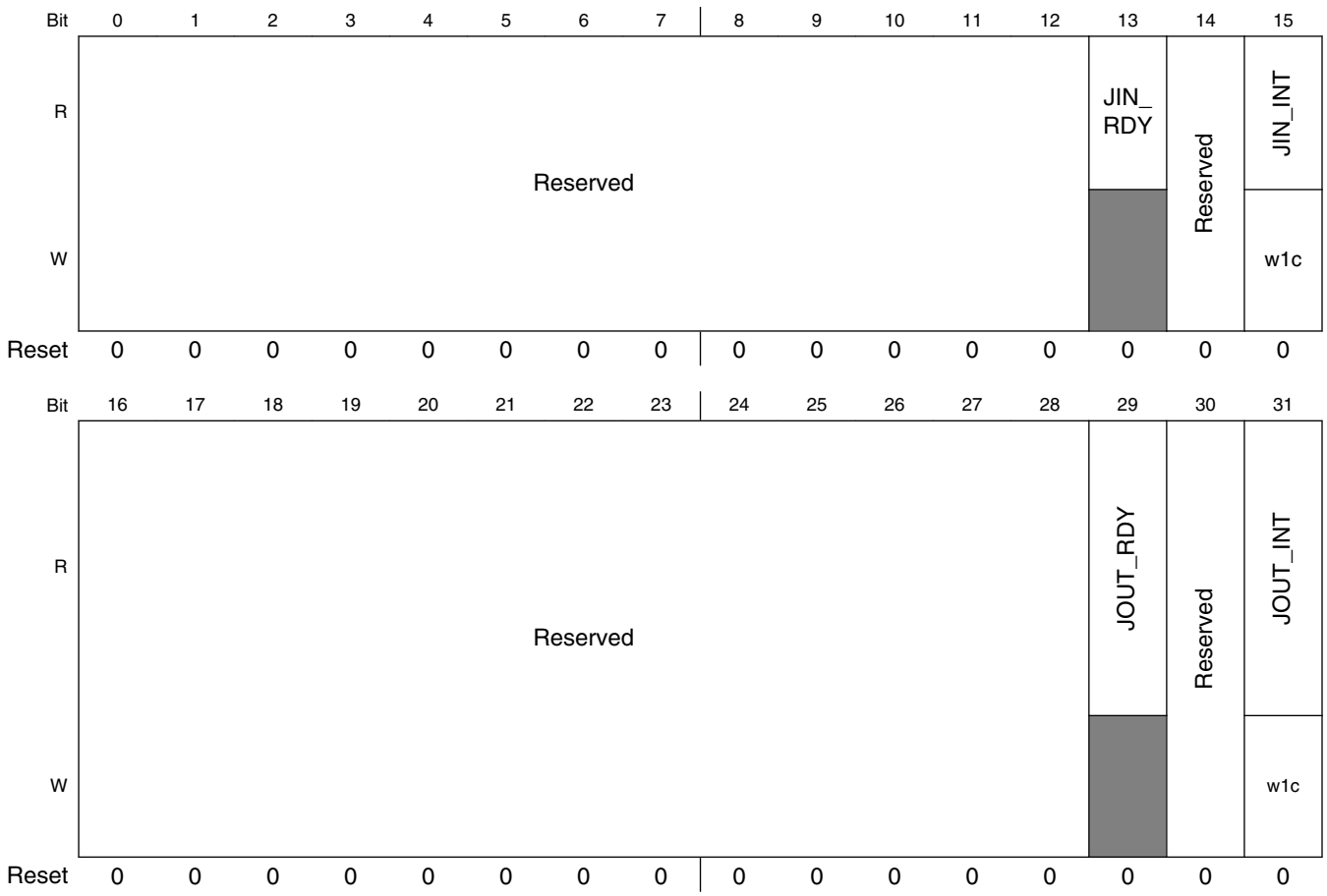
JDC_MCR field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 JIN_IEN	JIN Interrupt Enable. 0 Setting of MSR[JIN_INT] bit does not assert the JIN interrupt 1 Setting of MSR[JIN_INT] bit asserts the JIN interrupt
16–30 Reserved	This field is reserved.
31 JOUT_IEN	JOUT Interrupt Enable. 0 Setting of MSR[JOUT_INT] bit does not assert the JOUT interrupt 1 Setting of MSR[JOUT_INT] bit asserts the JOUT interrupt

71.3.2 Module Status Register (JDC_MSR)

The MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

Address: 0h base + 4h offset = 4h



JDC_MSR field descriptions

Field	Description
0–12 Reserved	This field is reserved.
13 JIN_RDY	JIN Ready (read only). 0 Cleared upon software read of JIN_IPS contents via IPS 1 Set when new data is written to the JIN_IPS register
14 Reserved	This field is reserved.
15 JIN_INT	JIN Interrupt.

Table continues on the next page...

JDC_MSR field descriptions (continued)

Field	Description
	0 Cleared by writing logic 1 1 Set when new data is written to the JIN_IPS register
16–28 Reserved	This field is reserved.
29 JOUT_RDY	JOUT Ready (read only). 0 Cleared upon tool read of JOUT register via JTAG port 1 Set when new data is written to the JOUT_IPS register
30 Reserved	This field is reserved.
31 JOUT_INT	JOUT Interrupt. 0 Cleared by writing logic 1 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register

71.3.3 JTAG Output Data Register (JDC_JOUT_IPS)

The JOUT_IPS register holds data written via IPS. The JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. This register is reset by system destructive reset.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JDC_JOUT_IPS field descriptions

Field	Description
0–31 Data	JOUT_IPS data.

71.3.4 JTAG Input Data Register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JIN_IPS register, where it can be read via IPS. This register is reset by system destructive reset.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Data																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JDC_JIN_IPS field descriptions

Field	Description
0–31 Data	JIN_IPS data.

71.4 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JIN_IPS register. The other JTAG register captures data from the JOUT_IPS register and ready status from the MSR to be shifted out via the JTAG port.

71.4.1 JTAG output data register (JOUT)

The JOUT register captures data from the JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JIN_RDY and JOUT_RDY status bits. This register is reset by system destructive reset and JTAG reset. The following figure shows the format of the JOUT register.

	33	…	2	1	0
R W	JOUT_IPS Data			JIN_RDY	JOUT_RDY
Reset	*[1]	*	*	*	*

1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in the following table.

Table 71-1. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from MSR
JOUT_RDY	State of JOUT_RDY bit from MSR

71.4.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JIN_IPS register to be read via IPS. This register is reset by system destructive reset and JTAG reset. The following figure shows the format of the JIN register.

	31	…	0
R W	JIN Data		
Reset	*[1]	*	*

1. The reset value of the JIN register is 32'b0.

The JIN register is described in the following table.

Table 71-2. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

71.5 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ

instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.

- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JIN_IPS register upon exit of the Update-DR state. An update of the JIN_IPS register sets the JIN_RDY flag bit, indicating new data is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

71.5.1 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

71.5.1.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 71-3. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

71.6 Use case example

A summary of how the JDC can be used to interact with the Hardware Security Module (HSM) to perform challenge/response password authorization is as follows:

1. After exit of reset, all registers are at their reset values. JIN_RDY bit value of 0 indicates the tool may write data to the JIN register. JOUT_RDY bit value of 0 indicates that software may write data to the JOUT_IPS register.
2. The MCR register is programmed to enable JOUT and JIN interrupts, or not.
3. Once the tool is ready to start the authorization process, it writes a value to the JIN register via execution of the JTAGC JIN_WRITE instruction.
4. Software monitors the state of the JIN_RDY bit or enables the JIN interrupt and uses the interrupt assertion as a trigger to start the authorization process.
5. Once the authorization process is started, software writes the first JOUT value to the JOUT_IPS register. This sets the JOUT_RDY bit value to 1.
6. Following the initial tool write to the JIN register, the tool polls the JOUT register to determine when software has provided a challenge word. Upon reading the JOUT register with JOUT_RDY bit set, the tool records the data value to be used to generate the appropriate response. The read of the JOUT register clears the JOUT_RDY bit. The clearing of the JOUT_RDY bit also sets the JOUT_INT bit and asserts the JOUT interrupt if the MCR[JOUT_IEN] is set.
7. With the JOUT_RDY bit cleared, software may now provide the next challenge word.
8. Once the tool has generated a response word, it uses the value of the JIN_RDY bit from the JOUT register read to determine when it can perform a write to the JIN register to provide the response.
9. The HSM can provide additional challenge words and read back the corresponding response words as needed, repeating the process.

Simply providing a password without using the challenge/response protocol can be done in a similar way. The tool provides 32-bits of the password at a time with each write to the JIN register, and monitors the JIN_RDY bit value of the JOUT register to determine when the next JIN register value can be written. There is no need for software to write the JOUT_IPS register in this case.

Chapter 72
Sequence Processing Unit (SPU)

72.1 Introduction

The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System level performance monitor functions are integrated into the SPU complex trigger logic, and thus allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients such as the cores, GTM, or crossbar switch, generate watchpoints and triggers when operating in their debug mode. The SPU collects these triggers (such as interrupt occurrence, address watchpoint, and so on) and uses them as conditions to sequence through states, with resultant actions (such as start/stop trace, start/stop counter, capture timebase, and so on) The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, complex logic-analyzer-like behavior can be achieved, providing vital real-time visibility and debug ability of the activities of the system.

72.1.1 SPU block diagram

The following figure shows a top-level block diagram of the SPU.

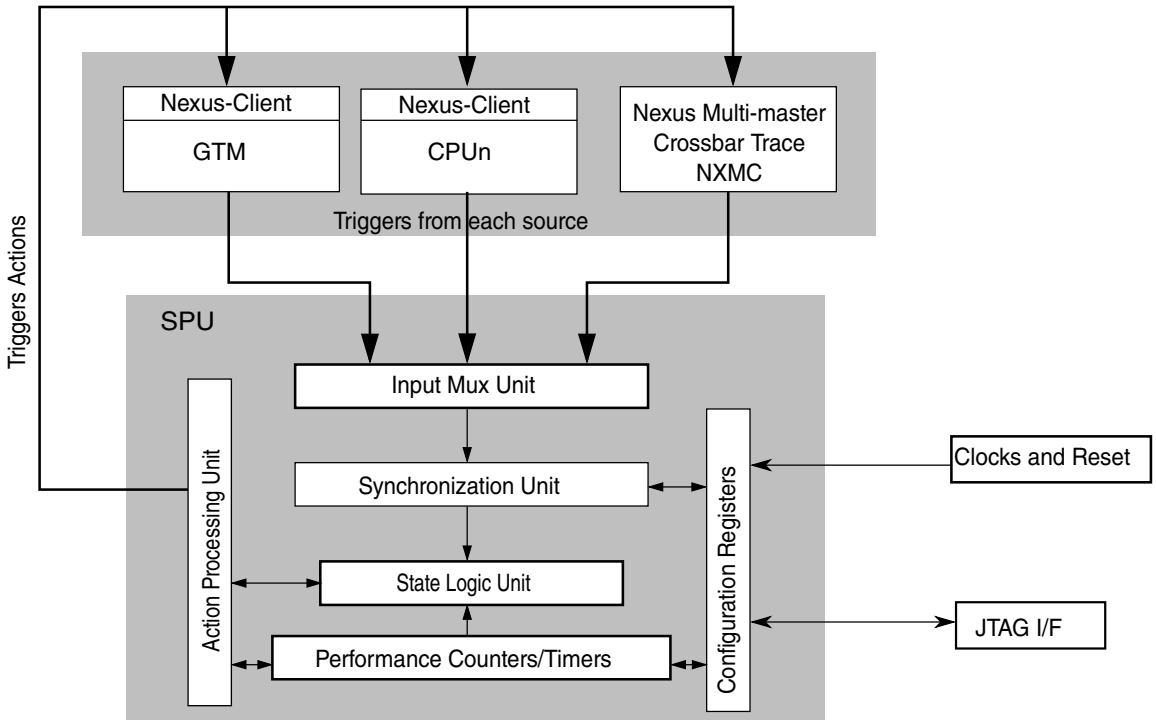


Figure 72-1. SPU block diagram

72.1.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs, GTM, and system bus clients) throughout the SoC. The SPU has the ability to create a state machine to trigger a debug action. Single or multiple actions can be triggered by a state machine, which results in debug events being created. There are a selection of timers and counters available to assist comparison events.

Complex trigger support is implemented in the State Logic Unit (SLU). There are numerous inputs coming from different clients. Users can choose 63 inputs from a set of 209 inputs at the level one Mux. There are actually 64 inputs to the Muxes but one of the inputs is permanently connected to a high level, so users can only choose 63 of the inputs. Out of these 63 inputs, users can select 16 inputs for each SLU. This is shown in Figure 72-4.

One SLU consists of four input AND gates, the output of which are ORed together. One of the inputs of each AND gate has the option to invert it. Also, the output of each AND gate can be optionally inverted. Finally, the output of each SLU has the option of inversion. The block diagram for one SLU state is shown in Figure 72-5. The SPU includes eight SLUs states.

Users can utilize these states to form a configurable state machine. This allows users to join states together with the if-then-else operations to create a sequence. Each state in a sequence has the ability to move to another state based on the true condition from the state logic, and the ability to route to another state based on the a false condition from the state logic. This is shown in Figure 72-6 and Figure 72-7. At a given time, a maximum of four sequences are allowed. Each sequence can have one state or maximum of eight states. Each state can only be used in one unique sequence. The number of states in all the sequences cannot exceed eight. Each state has the ability to trigger one to four actions based on a true/false condition in the active sequence.

An example of typical complex trigger for the SPU would be: if FunctionA is entered, monitor CPU0 writes to VariableB. If CPU0 writes a value to VariableB which is 0x100, then enable trace for CPU1 and halt CPU0.

The SPU consists of the following sub-blocks:

- Input Mux Unit—receives the numerous triggers from various clients and reduces them down into a user selected set. The muxing unit selects sixteen inputs for each sequencing SLU from a set of 63 static inputs. This selection is done using two levels of muxing. At the first level of muxing, users select 63 out of 209 inputs. At the second level of muxing, out of these 63 inputs, one input can be selected for each of the four input AND gates. There are 128 64 × 1 Muxes to cater the needs of 128 inputs of all the eight states. This input mux selection logic is shown in Figure 72-4.
- State Logic Units (SLU)—responsible for performing operations on the selected input events to produce desired events. Each SLU has a control register that manages how the various inputs are combined to produce a desired event. Only one input of each four input AND gate can be independently inverted. The combined event can also be optionally inverted. These optional inversions coupled with DeMorgan's law allows for generalized signal combinations. This is shown in Figure 72-5.
- Action Processing Unit—takes an action request from a SLU state and converts this into one or multiple actions. The action unit can be programmed to trigger various actions in response to each processed event from the SLU unit. The user can define the actions associated with each state from each SLU.
- Counters/Timers—The SPU implements 16 32-bit timer/counter functionality. Timer or counter selection is done via a configurable register in the SPU. Each counter/timer has its own control register. A 32-bit comparator value can optionally be compared against the timer/counter. These registers are accessible via JTAG.

72.1.3 Features

The features of the SPU are as follows:

- Clients:
  - 3 CPU(s)
  - 1 GTM
  - 2 System Bus clients (NXMC)
- Input watchpoints/triggers selection
- Input debug triggers for instruction execution selection
- Input interrupt selection
- Sequences and states criteria
  - Up to 4 user programmable number of active/simultaneous sequences
  - Up to 8 user programmable states that can be split between all four sequences
  - Each state can only be used in one unique sequence
  - A sequence can consist of a single state or any number of states up to 8
- State logic
  - Each state logic consist of sum of products (SOP)
  - SOP consist of four 4-input AND gates, and one 4-input OR gate
  - Optional inversion on one input of each AND gate of each state logic
  - Optional inversion on output of each AND gate.
  - Optional inversion on output of OR gate of each state
  - States are joined together with if-then-else statement to create a sequence
- SLU events
  - Up to 8 user programmable derived events/actions
  - Any state can optionally trigger of 1 to 4 actions based on a true condition
  - Any state can optionally trigger of 1 to 4 actions based on a false condition
  - User selection of trigger actions generated in response to each programmed trigger
- SPU counters/timers

    - 16 32-bit counters/timers
    - Timer/counter configurability selection via a register
    - Start/increment when used as timer/counter respectively
    - Reset when used as timer/counter respectively
    - Counter value compare match
    - Counter overflow event status
    - Generation of a greater than condition against a 32-bit comparator value (no support for range condition from single counter)
    - Range support using two counters
    - Use of counter/timer registers to generate counted or timed triggers
    - Use of counter/timer registers to capture system performance statistics
    - Each timer/counter value has the ability to be inserted into the Nexus trace stream upon an SPU action
    - Access to timer/counter values via JTAG
- Prescaler for timer mode only
    - Configurable prescaler of /1, /4, /16 or /32 of the clock for timer operation inside the SPU
- Asynchronous interfaces
    - SPU to/from CPU (fastest clock)
    - SPU to/from NXMC
    - SPU to/from GTM
    - SPU to/from NAR
    - SPU to/from DCI
    - SPU to/from JTAGM
- Interfaces
    - CPU
    - GTM
    - System bus

  - NAR
  - DCI
- Configuration registers—32-bit double word aligned configuration registers (byte, halfword access not supported)
- SPU actions—SPU block triggers various actions based on the derived events, these actions are defined in SPU actions.

72.2 SPU actions

Sequence actions describe what is triggered. A sequence state may only have a GoTo action to determine the next sequence state. The complete set of actions are as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Capture counter/timer values[0–15]—place the specified counter/timer value into the trace stream for counters[0–15]
- Halt1—single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- Halt2—single signal (EVTI1) that can be used to halt a pre-configured DCI group2
- Generate a watchpoint trigger for a respective ID field the NXMC (global timestamp must be enabled from the NAR)
- Capture the global timebase and place it into the trace stream with a watchpoint ID field of respective watchpoints at the NXMC (send TIMESTAMPx where x is related to an watchpoint ID field of NXMC only)
- Single interrupt for INTC to interrupt a CPU—de-assert (cleared) via JTAGM
- Generate a pulse on the EVTO pin—output pulse over EVTO1 and EVTO2
- Start or Stop Performance Counters (PMC) for a CPU source—Start/Stop CPUn
- Optional trace group 1 (Start/Stop Trace a pre-configured set of trace signals)—four (DTM,PTM,WTM,OTM) independent start/stop trace groups can be formed. One or more clients (CPUn and Nexus) can be part of a particular group.

Note

GoTo is not part of the Action list described above but it is implemented in the state logic to move from one state to another. If there is nothing defined for GoTo, it signifies the end of the sequence.

In the condition where a state (State n) updates a counter and the next state in the sequence (State n+1) evaluates the value of the same counter (for example, to compare if the counter is less-than a certain value), the update to the counter (from State n) is not applied before the counter evaluation in the next state (State n+1). This results in the old value of the counter being evaluated (in State n+1), rather than the desired updated value. This is due to hardware limitations of the SPU as the update to the counter does not complete until the clock cycle after the evaluation of the next state. This limitation is only applicable in Div1 mode and not in Div4 mode.

To workaround this hardware limitation, a delay should be inserted between the update to a counter and the evaluation of this counter. A dummy state can be used between the counter update (State n) and the counter evaluation (State n+1). A dummy state is a state with any input, that on both then/else conditions, the action proceeds to the next state (State n+1) where the counter evaluation occurs. The insertion of this dummy state provides sufficient time for the update of the counter to be applied, in time for the counter evaluation. The same outcome is also possible in a complex sequence with multiple states by scheduling states so that counter update and counter evaluation are not performed on consecutive states in a sequence.

72.3 SPU enable/disable mode

By default, the SPU is in disabled mode. In this state, all events/inputs are disabled such that no counting or other actions occur. All of the SPU control registers are accessible in this state. All of the configurations must be done before enabling the SPU by programming the SPU Enable/Disable register.

72.4 SPU module interface

The SPU is connected to many clients such as the cores, GTM, system bus, NAR and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in the following figure.

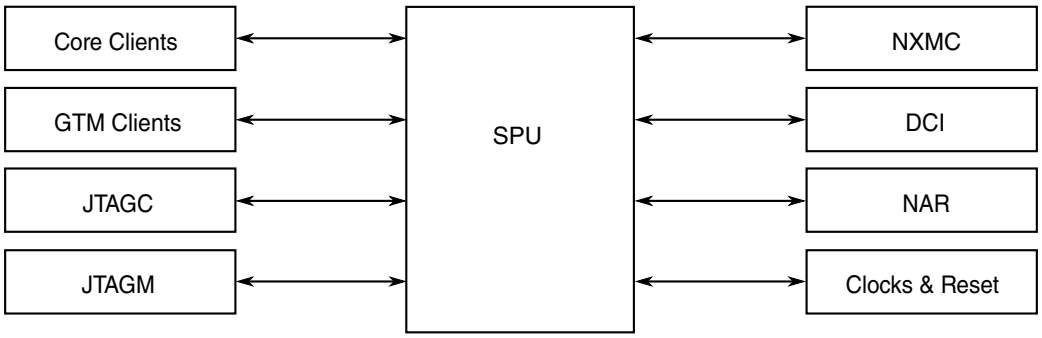


Figure 72-2. SPU module interface

72.5 Register definition

The SPU contains 32-bit aligned registers (byte and half-word access are not supported). All configuration registers in the SPU are JTAG registers such that these registers are accessible via JTAG interface only. Write access to reserved locations have no impact while read access to reserved locations always return 0. These registers have Nexus reset and JTAG reset control.

Note

The SPU does not check for correctness of the programmed values in the registers and programmers must ensure that correct values are being written.

The following table lists the memory mapped registers which control the operation of the SPU. There are two instructions for the SPU registers:

- Instruction to enable the SLU (first set of registers with offsets 0x1–0x49) registers is 4'h3
- Instruction to enable the Counter Control Unit (second set of registers with offsets 0x1–0x48) is 4'h2

Table 72-1. SPU register summary

Offset (hex)	Register	Access	Reset Value
1	Level1 Mux Selection 0 (L1SEL0)	R/W	32'h0
2	Level1 Mux Selection 1 (L1SEL1)	R/W	32'h0
3	Level1 Mux Selection 2 (L1SEL2)	R/W	32'h0
4	Level1 Mux Selection 3 (L1SEL3)	R/W	32'h0
5	Level1 Mux Selection 4(L1SEL4)	R/W	32'h0
6	Level1 Mux Selection 5 (L1SEL5)	R/W	32'h0

Table continues on the next page...

Table 72-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
7	Level1 Mux Selection 6 (L1SEL6)	R/W	32'h0
8	Level1 Mux Selection 7 (L1SEL7)	R/W	32'h0
9	Reserved for future use	—	—
A	Reserved for future use	—	—
B	Input Trigger Level Detection 2 (ITLD2)	R/W	32'h0
C	Input Trigger Level Detection 3 (ITLD3)	R/W	32'h0
D	CPU0 Processor Except Vector Prefix (C0PEVP)	R/W	32'h0
E	CPU1 Processor Except Vector Prefix (C1PEVP)	R/W	32'h0
F	CPU2 Processor Except Vector Prefix (C2PEVP)	R/W	32'h0
10	Reserved for future use	—	—
11	Reserved for future use	—	—
12	CPU0 Interrupt Priority Selection (C0PIS)	R/W	32'h0
13	CPU1 Interrupt Priority Selection (C1PIS)	R/W	32'h0
14	CPU2 Interrupt Priority Selection (C2PIS)	R/W	32'h0
15	Reserved for future use	—	—
16	Reserved for future use	—	—
17	Level2 Mux State0 Selection 0 (L20SEL0)	R/W	32'h0
18	Level2 Mux State0 Selection 1 (L20SEL1)	R/W	32'h0
19	Level2 Mux State0 Selection 2 (L20SEL2)	R/W	32'h0
1A	Level2 Mux State0 Selection 3 (L20SEL3)	R/W	32'h0
1B	Level2 Mux State1 Selection 0 (L21SEL0)	R/W	32'h0
1C	Level2 Mux State1 Selection 1 (L21SEL1)	R/W	32'h0
1D	Level2 Mux State1 Selection 2 (L21SEL2)	R/W	32'h0
1E	Level2 Mux State1 Selection 3 (L21SEL3)	R/W	32'h0
1F	Level2 Mux State2 Selection 0 (L22SEL0)	R/W	32'h0
20	Level2 Mux State2 Selection 1 (L22SEL1)	R/W	32'h0
21	Level2 Mux State2 Selection 2 (L22SEL2)	R/W	32'h0
22	Level2 Mux State2 Selection 3 (L22SEL3)	R/W	32'h0
23	Level2 Mux State3 Selection 0 (L23SEL0)	R/W	32'h0
24	Level2 Mux State3 Selection 1 (L23SEL1)	R/W	32'h0
25	Level2 Mux State3 Selection 2 (L23SEL2)	R/W	32'h0
26	Level2 Mux State3 Selection 3 (L23SEL3)	R/W	32'h0
27	Level2 Mux State4 Selection 0 (L24SEL0)	R/W	32'h0
28	Level2 Mux State4 Selection 1 (L24SEL1)	R/W	32'h0
29	Level2 Mux State4 Selection 2 (L24SEL2)	R/W	32'h0
2A	Level2 Mux State4 Selection 3 (L24SEL3)	R/W	32'h0
2B	Level2 Mux State5 Selection 0 (L25SEL0)	R/W	32'h0
2C	Level2 Mux State5 Selection 1 (L25SEL1)	R/W	32'h0
2D	Level2 Mux State5 Selection 2 (L25SEL2)	R/W	32'h0

Table continues on the next page...

Table 72-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
2E	Level2 Mux State6 Selection 3 (L25SEL3)	R/W	32'h0
2F	Level2 Mux State6 Selection 0 (L26SEL0)	R/W	32'h0
30	Level2 Mux State6 Selection 1 (L26SEL1)	R/W	32'h0
31	Level2 Mux State6 Selection 2 (L26SEL2)	R/W	32'h0
32	Level2 Mux State6 Selection 3 (L26SEL3)	R/W	32'h0
33	Level2 Mux State7 Selection 0 (L27SEL0)	R/W	32'h0
34	Level2 Mux State7 Selection 1 (L27SEL1)	R/W	32'h0
35	Level2 Mux State7 Selection 2 (L27SEL2)	R/W	32'h0
36	Level2 Mux State7 Selection 3 (L27SEL3)	R/W	32'h0
37	Input/Output Inversion Control State0 (IOIC0)	R/W	32'h0
38	Input/Output Inversion Control State1 (IOIC1)	R/W	32'h0
39	Input/Output Inversion Control State2 (IOIC2)	R/W	32'h0
3A	Input/Output Inversion Control State3 (IOIC3)	R/W	32'h0
3B	Input/Output Inversion Control State4 (IOIC4)	R/W	32'h0
3C	Input/Output Inversion Control State5 (IOIC5)	R/W	32'h0
3D	Input/Output Inversion Control State6 (IOIC6)	R/W	32'h0
3E	Input/Output Inversion Control State7 (IOIC7)	R/W	32'h0
3F	Sequence Control (SCTRL)	R/W	32'h0
40	State0 Goto Control (S0GC)	R/W	32'h0
41	State1 Goto Control (S1GC)	R/W	32'h0
42	State2 Goto Control (S2GC)	R/W	32'h0
43	State3 Goto Control (S3GC)	R/W	32'h0
44	State4 Goto Control (S4GC)	R/W	32'h0
45	State5 Goto Control (S5GC)	R/W	32'h0
46	State6 Goto Control (S6GC)	R/W	32'h0
47	State7 Goto Control (S7GC)	R/W	32'h0
48	SLU Status (SS)	R/W	32'h0
49	SPU Enable (SE)	R/W	32'h0
1	State0 True Action 0 (S0TA0)	R/W	32'h0
2	State0 True Action 1 (S0TA1)	R/W	32'h0
3	State1 True Action 0 (S1TA0)	R/W	32'h0
4	State1 True Action 1 (S1TA1)	R/W	32'h0
5	State2 True Action 0 (S2TA0)	R/W	32'h0
6	State2 True Action 1 (S2TA1)	R/W	32'h0
7	State3 True Action 0 (S3TA0)	R/W	32'h0
8	State3 True Action 1 (S3TA1)	R/W	32'h0
9	State4 True Action 0 (S4TA0)	R/W	32'h0
A	State4 True Action 1 (S4TA1)	R/W	32'h0
B	State5 True Action 0 (S5TA0)	R/W	32'h0

Table continues on the next page...

Table 72-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
C	State5 True Action 1(S5TA1)	R/W	32'h0
D	State6 True Action 0 (S6TA0)	R/W	32'h0
E	State6 True Action 1(S6TA1)	R/W	32'h0
F	State7 True Action 0 (S7TA0)	R/W	32'h0
10	State7 True Action 1(S7TA1)	R/W	32'h0
11	State0 False Action 0 (S0FA0)	R/W	32'h0
12	State0 False Action 1 (S0FA1)	R/W	32'h0
13	State1 False Action 0 (S1FA0)	R/W	32'h0
14	State1 False Action 1(S1FA1)	R/W	32'h0
15	State2 False Action 0 (S2FA0)	R/W	32'h0
16	State2 False Action 1(S2FA1)	R/W	32'h0
17	State3 False Action 0 (S3FA0)	R/W	32'h0
18	State3 False Action 1(S3FA0)	R/W	32'h0
19	State4 False Action 0 (S4FA0)	R/W	32'h0
1A	State4 False Action 1 (S4FA1)	R/W	32'h0
1B	State5 False Action 0 (S5FA0)	R/W	32'h0
1C	State5 False Action 1 (S5FA1)	R/W	32'h0
1D	State6 False Action 0 (S6FA0)	R/W	32'h0
1E	State6 False Action 1 (S6FA1)	R/W	32'h0
1F	State7 False Action 0 (S7FA0)	R/W	32'h0
20	State7 False Action 1 (S7FA1)	R/W	32'h0
21	DTM Trace Group Configuration (DTMTGC)	R/W	32'h0
22	PTM Trace Group Configuration (PTMTGC)	R/W	32'h0
23	OTM Trace Group Configuration (OTMTGC)	R/W	32'h0
24	WTM Trace Group Configuration (WTMTGC)	R/W	32'h0
25	Interrupt Status (INTS)	R/W	32'h0
26	Counter Control 0 (CCTRL0)	R/W	32'h0
27	Counter Control 1 (CCTRL1)	R/W	32'h0
28	Counter Control 2 (CCTRL2)	R/W	32'h0
29	Counter Control 3 (CCTRL3)	R/W	32'h0
2A	Counter Control 4 (CCTRL4)	R/W	32'h0
2B	Counter Control 5 (CCTRL5)	R/W	32'h0
2C	Counter Control 6 (CCTRL6)	R/W	32'h0
2D	Counter Control 7 (CCTRL7)	R/W	32'h0
2E	Counter Control 8 (CCTRL8)	R/W	32'h0
2F	Counter Control 9 (CCTRL9)	R/W	32'h0
30	Counter Control 10 (CCTRL10)	R/W	32'h0
31	Counter Control 11 (CCTRL11)	R/W	32'h0
32	Counter Control 12 (CCTRL12)	R/W	32'h0

Table continues on the next page...

Table 72-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
33	Counter Control 13 (CCTRL13)	R/W	32'h0
34	Counter Control 14 (CCTRL14)	R/W	32'h0
35	Counter Control 15 (CCTRL15)	R/W	32'h0
36	Counter Compare 0 (CCMP0)	R/W	32'h0
37	Counter Compare 1 (CCMP1)	R/W	32'h0
38	Counter Compare 2 (CCMP2)	R/W	32'h0
39	Counter Compare 3 (CCMP3)	R/W	32'h0
3A	Counter Compare 4 (CCMP4)	R/W	32'h0
3B	Counter Compare 5 (CCMP5)	R/W	32'h0
3C	Counter Compare 6 (CCMP6)	R/W	32'h0
3D	Counter Compare 7 (CCMP7)	R/W	32'h0
3E	Counter Compare 8 (CCMP8)	R/W	32'h0
3F	Counter Compare 9 (CCMP9)	R/W	32'h0
40	Counter Compare 10 (CCMP10)	R/W	32'h0
41	Counter Compare 11 (CCMP11)	R/W	32'h0
42	Counter Compare 12 (CCMP12)	R/W	32'h0
43	Counter Compare 13 (CCMP13)	R/W	32'h0
44	Counter Compare 14 (CCMP14)	R/W	32'h0
45	Counter Compare 15 (CCMP15)	R/W	32'h0
46	Counter Compare Status (CCOMS)	R/W	32'h0
47	Counter Overflow Status (COS)	R/W	32'h0
48	Counter Capture Status (CCAPS)	R/W	32'h0

72.5.1 Level1 input Mux configurations registers

A set of input Mux control registers is used to narrow down the large number of SPU watchpoints to a manageable set. There are sixty-four 8 × 1 (eight inputs and one output) multiplexers. Three bits per Mux are used to select one input out of 8 inputs. 8 Muxes are configured with each 32-bit register; in other words, 8 out of 64 inputs for each register. This provides the flexibility to the programmer to route any input trigger to any state input. The detail of this selection is shown in Figure 72-4.

Note

Users cannot program MUX 63 in L1SEL7 (the 64th mux). All eight inputs of the 63rd mux are tied to high and the output is always high (1'b1).

72.5.1.1 Level1 Mux Selection 0 (L1SEL0)

The following table shows the format of the L1SEL0 register. The values for the fields of the L1SEL0 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x01 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 7			0	MUX 6			0	MUX 5			0	MUX 4			0	MUX 3			0	MUX 2			0	MUX 1			0	MUX 0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.2 Level1 Mux Selection 1 (L1SEL1)

The following table shows the format of the L1SEL1 register. The values for the fields of the L1SEL1 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x02 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 15			0	MUX 14			0	MUX 13			0	MUX 12			0	MUX 11			0	MUX 10			0	MUX 9			0	MUX 8		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.3 Level1 Mux Selection 2 (L1SEL2)

The following table shows the format of the L1SEL2 register. The values for the fields of the L1SEL2 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x03 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 23			0	MUX 22			0	MUX 21			0	MUX 20			0	MUX 19			0	MUX 18			0	MUX 17			0	MUX 16		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.4 Level1 Mux Selection 3 (L1SEL3)

The following table shows the format of the L1SEL3 register. The values for the fields of the L1SEL3 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x04 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 31			0	MUX 30			0	MUX 29			0	MUX 28			0	MUX 27			0	MUX 26			0	MUX 25			0	MUX 24		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.5 Level1 Mux Selection 4 (L1SEL4)

The following table shows the format of the L1SEL4 register.The values for the fields of the L1SEL4 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x05 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 39			0	MUX 38			0	MUX 37			0	MUX 36			0	MUX 35			0	MUX 34			0	MUX 33			0	MUX 32		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.6 Level1 Mux Selection 5 (L1SEL5)

The following table shows the format of the L1SEL5 register. The values for the fields of the L1SEL5 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x06 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 47			0	MUX 46			0	MUX 45			0	MUX 44			0	MUX 43			0	MUX 42			0	MUX 41			0	MUX 40		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.7 Level1 Mux Selection 6 (L1SEL6)

The following table shows the format of the L1SEL6 register.The values for the fields of the L1SEL6 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x07 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 55			0	MUX 54			0	MUX 53			0	MUX 52			0	MUX 51			0	MUX 50			0	MUX 49			0	MUX 48		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.8 Level1 Mux Selection 7 (L1SEL7)

The following table shows the format of the L1SEL7 register. The values for the fields of the L1SEL7 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured. All eight inputs of MUX 63 are tied to logic high (1'b1) and output is always high.

Offset: 0x08 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 63			0	MUX 62			0	MUX 61			0	MUX 60			0	MUX 59			0	MUX 58			0	MUX 57			0	MUX 56		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.9 Input triggers level detection control 2 (ITLD2)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. The following table shows the format of the ITLD2 register. All of the MUX fields in the ITLD2 register have the same description as shown in Table 72-2.

Offset 0x0B Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
W																

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The ITLD registers Mux field is described in the following table.

Table 72-2. ITLD register field descriptions

Field	Description
Mux*	Mux Output Signal Detection. 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

72.5.1.10 Input triggers level detection control 3 (ITLD3)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. The following table shows the format of the ITLD3 register. All of the MUX fields in the ITLD3 register have the same description as shown in Table 72-2.

Offset 0x0C Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Mux63		Mux62		Mux61		Mux60		Mux59		Mux58		Mux57		Mux56	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Mux55		Mux54		Mux53		Mux52		Mux51		Mux50		Mux49		Mux48	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

72.5.1.11 CPU0 processor exception vector prefix (C0PEVP)

The SPU captures the exception vector from CPU0 when the exception enable signal is asserted. The exception vector decoding is defined in Table 72-30. The SPU compares this captured vector with the value in the C0PEVP register. The following table shows the format of the C0PEVP register.

Offset: 0x0D Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU0															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0PEVP register is described in the following table.

Table 72-3. C0PEVP register field descriptions

Field	Description
15:0 CPU0	CPU0 processor exception vector. Program this field with one of the vector values from Table 72-30 to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in Table 72-30.
31:16	Reserved

72.5.1.12 CPU1 processor exception vector prefix (C1PEVP)

The SPU captures the exception vector from CPU1 when the exception enable signal is asserted. The exception vector decoding is defined in Table 72-30. The SPU compares this captured vector with the value in the C1PEVP register. The following table shows the format of the C1PEVP register.

Offset: 0x0E Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU1															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C1PEVP register is described in the following table.

Table 72-4. C1PEVP register field descriptions

Field	Description
15:0 CPU1	CPU1 processor exception vector. Program this field with one of the vector values from Table 72-30 to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in Table 72-30.
31:16	Reserved

72.5.1.13 CPU2 processor exception vector prefix (C2PEVP)

The SPU captures the exception vector from CPU2 when the exception enable signal is asserted. The exception vector decoding is defined in the following table. The SPU compares this captured vector with the value in the C2PEVP register. The following table shows the format of the C2PEVP register.

Offset: 0x0F Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C2PEVP register is described in the following table.

Table 72-5. C2PEVP register field descriptions

Field	Description
15:0 CPU2	CPU2 processor exception vector. Program this field with one of the vector values from Table 72-30 to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in Table 72-30.
31:16	Reserved

72.5.1.14 CPU0 interrupt priority selection (C0PIS)

The following table shows the format of the C0PIS register. The SPU compares the input CPR vector from CPU0 with the value programmed to the C0PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x012 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU0							
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0PIS register is described in the following table.

Table 72-6. C0PIS register field descriptions

Field	Description
7:0 CPU0	CPU0 Priority Interrupt Selection.
31:8	Reserved

72.5.1.15 CPU1 interrupt priority selection (C1PIS)

The following table shows the format of the C1PIS register. The SPU compares the input CPR vector from CPU1 with the value programmed to the C1PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x013 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU1							
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C1PIS register is described in the following table.

Table 72-7. C1PIS register field descriptions

Field	Description
7:0 CPU1	CPU1 Priority Interrupt Selection.
31:8	Reserved

72.5.1.16 CPU2 interrupt priority selection (C2PIS)

The following table shows the format of the C2PIS register. The SPU compares the input CPR vector from CPU2 with the value programmed to the C2PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x014 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2							
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C2PIS register is described in the following table.

Table 72-8. C2PIS register field descriptions

Field	Description
7:0 CPU2	CPU2 Priority Interrupt Selection.
31:8	Reserved

72.5.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in Level1 input Mux configurations registers) narrows down the large number of SPU watchpoints to a manageable set using sixty-four 8 × 1 multiplexers. At Level2, 16 inputs are selected for each SLU from the 64 level1 outputs using 16 64 × 1 multiplexers. Six bits per Mux select one input out of 64 (in fact 63) inputs. Each 32-bit register configures the AND gates of one state; therefore, eight registers are used to configure inputs for the 8 SLUs. All the 63 inputs are static for all the eight states. Any 16 inputs can be selected from the 63 input triggers for a state.

The detail selection is shown in the following figure for State0. Similar logic is provided for States 1–7.

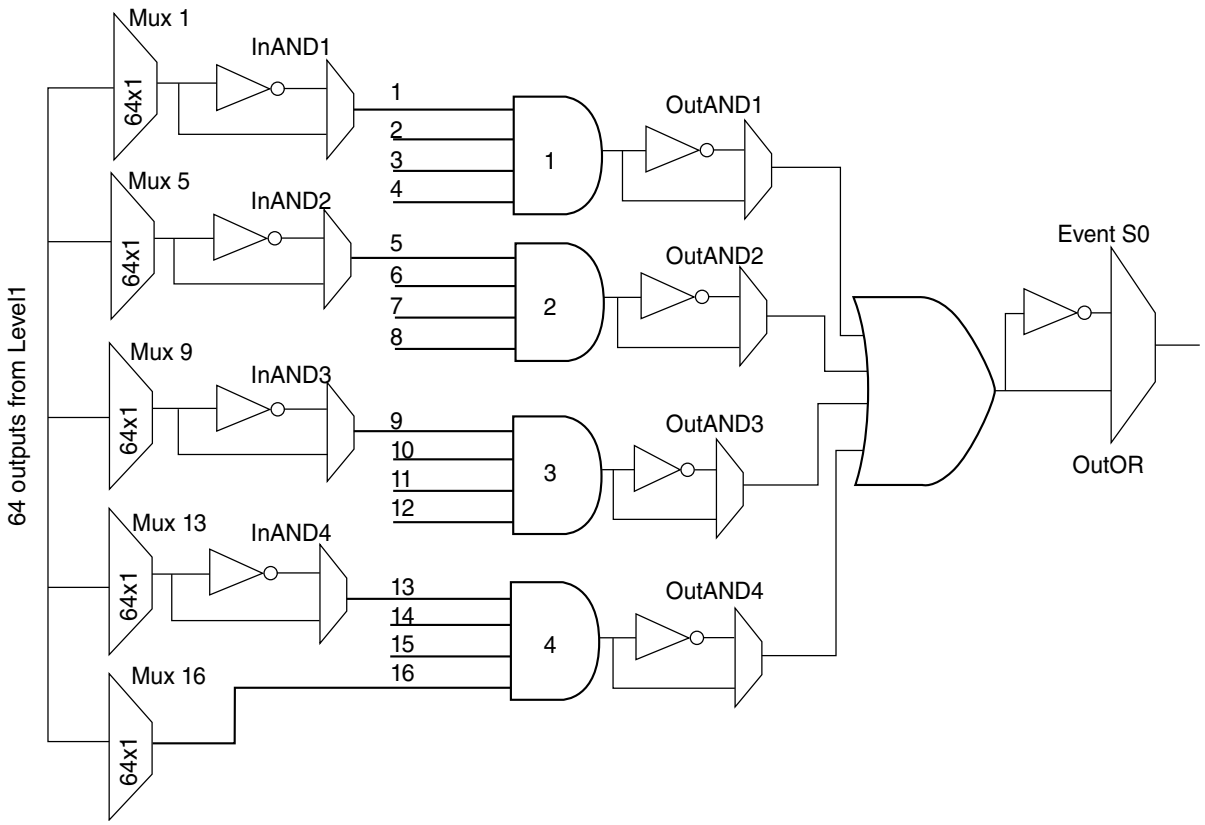


Figure 72-3. Level2 Mux AND gates input pin routing for State0

Note

If any SLU AND gate is unused, then its output is logic low (1'b0). Thus, if all four inputs of a particular AND gate are at the default value 1'b1 (none of the inputs have been routed from the Level 2 mux to the inputs of AND gates), then the output of that AND gate is logic low. This provides protection to avoid the masking of other inputs to the OR gate.

72.5.2.1 Level2 Mux state n selection 0 (L2nSEL0)

The following table shows the format of the L2nSEL0 register, where n, the state number, is equal to 0–7.

Offset:	0x17 (State0) 0x1B (State1) 0x1F (State2) 0x23 (State3) 0x27 (State4) 0x2B (State5)	Access: User read/write

Table continues on the next page...

0x2F (State6)

0x33 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FirstANDInput4						0	0	FirstANDInput3						0	0	FirstANDInput2						0	0	FirstANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL0 register fields are described in the following table.

Table 72-9. L2nSEL0 register field descriptions

Field	Description	
5:0	First AND Gate Input1 Selection.	
FirstANDInput1	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
7:6	Reserved. Read returns 0.	
13:8	First AND Gate Input 2 Selection.	
FirstANDInput2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
15:14	Reserved. Read returns 0.	
21:16	First AND Gate Input 3 Selection.	
ANDInput3	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
23:22	Reserved. Read returns 0.	
29:24	First AND Gate Input 4 Selection.	
FirstANDInput4	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output

Table continues on the next page...

Table 72-9. L2nSEL0 register field descriptions (continued)

Field	Description
	000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
31:30	Reserved. Read returns 0.

72.5.2.2 Level2 Mux state n selection 1 (L2nSEL1)

The following table shows the format of the L2nSEL1 register, where n, the state number, is equal to 0–7.

Offset: 0x18 (State0), 0x1C (State1), 0x20 (State2), 0x24 (State3), 0x28 (State4), 0x2C (State5), 0x30 (State6), 0x34 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	SecondANDInput4						0	0	SecondANDInput3						0	0	SecondANDInput2						0	0	SecondANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL1 register fields are described in the following table.

Table 72-10. L2nSEL1 register field descriptions

Field	Description
5:0 SecondANDInput1	Second AND Gate Input1 Selection. 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
7:6	Reserved. Read returns 0.

Table continues on the next page...

Table 72-10. L2nSEL1 register field descriptions (continued)

Field	Description
13:8 SecondANDInput 2	Second AND Gate Input 2 Selection. 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
15:14	Reserved. Read returns 0.
21:16 SecondANDInput 3	Second AND Gate Input 3 Selection. 000000 No Input is selected (tie to 1'b1) 00000 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
23:22	Reserved. Read returns 0.
29:24 SecondANDInput 4	Second AND Gate Input 4 Selection. 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
31:30	Reserved. Read returns 0.

72.5.2.3 Level2 Mux state n selection 2 (L2nSEL2)

The following table shows the format of the L2nSEL2 register, where n, the state number, is equal to 0–7.

Offset: 0x19 (State0), 0x1D (State1), 0x25 (State3), 0x21 (State2), 0x29 (State4), 0x2D (State5) — Access: User read/write

Table continues on the next page...

0x31 (State6)

0x35 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	ThirdANDInput4						0	0	ThirdANDInput3						0	0	ThirdANDInput2						0	0	ThirdANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL2 register fields are described in the following table.

Table 72-11. L2nSEL2 register field descriptions

Field	Description	
5:0	Third AND Gate Input1 Selection.	
ThirdANDInput1	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
7:6	Reserved. Read returns 0.	
13:8	Third AND Gate Input 2 Selection.	
ThirdANDInput2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
15:14	Reserved. Read returns 0.	
21:16	Third AND Gate Input 3 Selection.	
ThirdANDInput3	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
23:22	Reserved. Read returns 0.	
29:24	Third AND Gate Input 4 Selection.	
ThirdANDInput4	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output

Table continues on the next page...

Table 72-11. L2nSEL2 register field descriptions (continued)

Field	Description
	000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
31:30	Reserved. Read returns 0.

72.5.2.4 Level2 Mux state n selection 3 (L2nSEL3)

The following table shows the format of the L2nSEL3 register, where n, the state number, is equal to 0–7.

Offset: 0x1A (State0), 0x1E (State1), 0x22 (State2), 0x26 (State3), 0x2A (State4), 0x2E (State5), 0x32 (State6), 0x36 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FourthANDInput4						0	0	FourthANDInput3						0	0	FourthANDInput2						0	0	FourthANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL3 register fields are described in the following table.

Table 72-12. L2nSEL3 register field descriptions

Field	Description
5:0	Fourth AND Gate Input1 Selection.
FourthANDInput1	000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 000010 Level1 MUX 1 000011 Level1 MUX 2 … 111110 Level1 MUX 61 111111 Level1 MUX 62
7:6	Reserved. Read returns 0.

Table continues on the next page...

Table 72-12. L2nSEL3 register field descriptions (continued)

Field	Description	
13:8	Fourth AND Gate Input 2 Selection.	
FourthANDInput2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
15:14	Reserved. Read returns 0.	
21:16	Fourth AND Gate Input 3 Selection.	
FourthANDInput3	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
23:22	Reserved. Read returns 0.	
29:24	Fourth AND Gate Input 4 Selection.	
FourthANDInput4	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
31:30	Reserved. Read returns 0.	

72.5.3 Sequence unit registers

72.5.3.1 Input/output Inversion control state n (IOICn)

An input control register is provided for each event generated by the SLU. This register is programmed to specify how the input signals to each SLU are combined to generate the resulting event for that particular sequence. In total, there are 16 inputs for each state and only one (the first) of the inputs can be used with or without inversion. The input can be left unused if not required for that state by selecting 000000 (No input selected option) in

the L2nSEL registers. The output of each AND gate has the optional inversion control as well. The output can be routed directly or with inversion from each AND gate to the input of OR gate. The following table shows the format of the IOICn register, where n, the state number, is equal to 0–7.

Offset 0x37 (State0) 0x38 (State1) 0x39 (State2) 0x3A (State3) 0x3B (State4) 0x3C (State5) 0x3D (State6) 0x3E (State7) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	OutOR	OutAND 4	OutAND 3	OutAND 2	OutAND 1	InAND4	InAND3	InAND2	InAND1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The IOICn register fields are described in the following table.

Table 72-13. IOICn register field descriptions

Field	Description
0 InAND1	Input 1 AND Gate 1 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
1 InAND2	Input 1 AND Gate 2 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
2 InAND3	Input 1 AND Gate 3 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
3 InAND4	Input 1 AND Gate 4 Control. 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
4 OutAND1	Output AND Gate 1 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event

Table continues on the next page...

Table 72-13. IOICn register field descriptions (continued)

Field	Description
5 OutAND2	Output AND Gate 2 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
6 OutAND3	Output AND Gate 3 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
7 OutAND4	Output of AND Gate 4 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
8 OutOR	Output OR GATE Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
9:31	Reserved. Read returns 0.

72.5.3.2 Sequence control (SCTRL)

This configuration register is provided to enable the sequence and to select the states that are part of a particular sequence. There can be at most four active sequences. Each sequence can have one state or a maximum of eight states. However, each state can only be used in one unique sequence. In other words, if one state has been selected for a sequence, then this state cannot be used for any other defined sequence. Each sequence has a start state and end state. The following table shows the format of the SCTRL register.

Offset 0x3F Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SCTRL register fields are described in the following table.

Table 72-14. SCTRL register field descriptions

Field	Description
0 ST0	Sequence1 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 1. 0 Sequence 1 is disabled 1 Sequence 1 is enabled
3:1 ST1	Sequence 1 Start State. 000 State 0 (Always)
6:4 ST2	Sequence 1 End State. 000 State 0 is the Stop state for sequence 1 001 State 1 is the Stop state for sequence 1 010 State 2 is the Stop state for sequence 1 011 State 3 is the Stop state for sequence 1 100 State 4 is the Stop state for sequence 1 101 State 5 is the Stop state for sequence 1 110 State 6 is the Stop state for sequence 1 111 State 7 is the Stop state for sequence 1
7	Reserved. Read returns 0.
8 ST0	Sequence2 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 2. 0 Sequence 2 is disabled 1 Sequence 2 is enabled
11:9 ST1	Sequence 2 Start State.[2] 000 State 0 is the Start state for sequence 2 001 State 1 is the Start state for sequence 2 010 State 2 is the Start state for sequence 2 011 State 3 is the Start state for sequence 2 100 State 4 is the Start state for sequence 2 101 State 5 is the Start state for sequence 2 110 State 6 is the Start state for sequence 2 111 State 7 is the Start state for sequence 2
14:12 ST2	Sequence 2 End State. 000 State 0 is the Stop state for sequence 2 001 State 1 is the Stop state for sequence 2 010 State 2 is the Stop state for sequence 2 011 State 3 is the Stop state for sequence 2 100 State 4 is the Stop state for sequence 2 101 State 5 is the Stop state for sequence 2 110 State 6 is the Stop state for sequence 2 111 State 7 is the Stop state for sequence 2
15	Reserved. Read returns 0.
16 ST0	Sequence 3 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 3. 0 Sequence 3 is disabled 1 Sequence 3 is enabled

Table continues on the next page...

Table 72-14. SCTRL register field descriptions (continued)

Field	Description
19:17	Sequence 3 Start State.[2]
ST1	000 State 0 is the Start state for sequence 3
	001 State 1 is the Start state for sequence 3
	010 State 2 is the Start state for sequence 3
	011 State 3 is the Start state for sequence 3
	100 State 4 is the Start state for sequence 3
	101 State 5 is the Start state for sequence 3
	110 State 6 is the Start state for sequence 3
	111 State 7 is the Start state for sequence 3
22:20	Sequence 3 End State.
ST2	000 State 0 is the Stop state for sequence 3
	001 State 1 is the Stop state for sequence 3
	010 State 2 is the Stop state for sequence 3
	011 State 3 is the Stop state for sequence 3
	100 State 4 is the Stop state for sequence 3
	101 State 5 is the Stop state for sequence 3
	110 State 6 is the Stop state for sequence 3
	111 State 7 is the Stop state for sequence 3
23	Reserved. Read returns 0.
24	Sequence 4 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 4.
ST0	0 Sequence 4 is disabled
	1 Sequence 4 is enabled
27:25	Sequence 4 Start State.[2]
ST1	000 State 0 is the Start state for sequence 4
	001 State 1 is the Start state for sequence 4
	010 State 2 is the Start state for sequence 4
	011 State 3 is the Start state for sequence 4
	100 State 4 is the Start state for sequence 4
	101 State 5 is the Start state for sequence 4
	110 State 6 is the Start state for sequence 4
	111 State 7 is the Start state for sequence 4
30:28	Sequence 4 End State.
ST2	000 State 0 is the Stop state for sequence 4
	001 State 1 is the Stop state for sequence 4
	010 State 2 is the Stop state for sequence 4
	011 State 3 is the Stop state for sequence 4
	100 State 4 is the Stop state for sequence 4
	101 State 5 is the Stop state for sequence 4
	110 State 6 is the Stop state for sequence 4
	111 State 7 is the Stop state for sequence 4

Table continues on the next page...

Table 72-14. SCTRL register field descriptions (continued)

Field	Description
31	Reserved. Read returns 0.

1. Multiple active sequences such as Sequence 2, Sequence 3 and Sequence 4 must be programmed with Div4 option only.
2. Ideally, the start state for a sequence should be the state immediately after the stop state for the previous sequence. For example, if sequence 1 starts at state 0 and ends at state3, sequence 2 should start at state4.

72.5.3.3 State n goto control (SnGC)

SLU states are linked to each other based on if-then-else conditions. There are two conditions: true and false. A SLU state can remain in the same state or jump to any other state. The state goto control register is programmed to specify the GoTo a particular state from current state.

The following table shows the format of the SnGC register, where n, the state number, is equal to 0–7.

Offset: 0x40 (State0), 0x41 (State1), 0x42 (State2), 0x43 (State3), 0x44 (State4), 0x45 (State5), 0x46 (State6), 0x47 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NF			FE	NT			TE
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnGC register fields are described in the following table.

Table 72-15. SnGC register field descriptions

Field		Description
0		GoTo Control of State Under True Condition. Defines the Goto control of a state under the true condition. If this bit is enabled, then the State moves to the next state based on bits[3:1]. If disabled, then this is the end of a sequence when the sequence comes to this state. Actions associated with states under true condition are executed. Disabling this bit ends the sequence under a true condition.
TE	0	Disable Goto control
	1	Enable Goto control

Table continues on the next page...

Table 72-15. SnGC register field descriptions (continued)

Field	Description	
3:1	Next State on True Condition. Defines the Goto next state in case of true condition from the present state. States may remain in the same state on a true condition if the same state is chosen as the next state.	
NT	000	Goto State 0
	001	Goto State 1
	010	Goto State 2
	011	Goto State 3
	100	Goto State 4
	101	Goto State 5
	110	Goto State 6
	111	Goto State 7
4	GOTO Control of State Under False Condition. Defines the Goto control of a state under the false condition. If this bit is enabled, then the state moves to the next state based on bits[7:5] . If disabled, then this is the is end of a sequence when the sequence comes to this state. Actions associated with states under false condition are executed. Disabling this bit end the sequence under a false condition.	
FE	0	Disable Goto control
	1	Enable Goto control
7:5	Next State on False Condition. Defines the Goto next state in case of false condition from the present state. States may remain in the same state on a false condition if the same state is chosen as the next state.	
NF	000	Goto State 0
	001	Goto State 1
	010	Goto State 2
	011	Goto State 3
	100	Goto State 4
	101	Goto State 5
	110	Goto State 6
	111	Goto State 7
31:8	Reserved. Read returns 0.	

72.5.4 SLU status (SS) register

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. The following table shows the format of the SS register.

Offset 0x48 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	IM2	IM1	IM0	EM2	EM1	EM0	SAS4			SAS3
W							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAS3		SAS2			SAS1			SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SS register fields are described in the following table.

Table 72-16. SS register field descriptions

Field	Description
0 SH0	State Hit 0. This bit is set when State0 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
1 SH1	State Hit 1. This bit is set when State1 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
2 SH2	State Hit 2. This bit is set when State2 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
3 SH3	State Hit 3. This bit is set when State3 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
4 SH4	State Hit 4. This bit is set when State4 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
5 SH5	State Hit 5. This bit is set when State5 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
6 SH6	State Hit 6. This bit is set when State6 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
7 SH7	State Hit 7. This bit is set when State7 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
10 :8 SAS1	Sequence 1 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 1 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state

Table continues on the next page...

Table 72-16. SS register field descriptions (continued)

Field	Description
	110 State6 is the active state
	111 State7 is the active state
13 :11	Sequence 2 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 2 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.
SAS2	000 State0 is the active state
	001 State1 is the active state
	010 State2 is the active state
	011 State3 is the active state
	100 State4 is the active state
	101 State5 is the active state
	110 State6 is the active state
	111 State7 is the active state
16:14	Sequence 3 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 3 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.
SAS3	000 State0 is the active state
	001 State1 is the active state
	010 State2 is the active state
	011 State3 is the active state
	100 State4 is the active state
	101 State5 is the active state
	110 State6 is the active state
	111 State7 is the active state
19 :17	Sequence 4 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 4 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.
SAS4	000 State0 is the active state
	001 State1 is the active state
	010 State2 is the active state
	011 State3 is the active state
	100 State4 is the active state
	101 State5 is the active state
	110 State6 is the active state
	111 State7 is the active state
20	Exception Match 0. This bit indicates the status of the processor exception vector match value of CPU0.
EM0	0 No match
	1 Match occurred
21	Exception Match 1. This bit indicates the status of the processor exception vector match value of CPU1.
EM1	0 No match
	1 Match occured
22	Exception Match 2. This bit indicates the status of the processor exception vector match value of CPU2.

Table continues on the next page...

Table 72-16. SS register field descriptions (continued)

Field	Description
EM2	0 No match 1 Match occurred
23 IM0	Interrupt Match 0. This bit indicates the status of the match value on interrupt priority register of CPU0. 0 No match 1 Match occurred
24 IM1	Interrupt Match 1. This bit indicates the status of the match value on interrupt priority register of CPU1. 0 No match 1 Match occurred
25 IM2	Interrupt Match 2. This bit indicates the status of the match value on interrupt priority register of CPU2. 0 No match 1 Match occurred
31:26	Reserved. Read returns 0

72.5.5 SPU enable (SE) register

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. The following table shows the format of the SE register.

Offset: 0x49 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV	SED
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SE register fields are described in the following table.

Table 72-17. SE register field descriptions

Field	Description
0 SED	SPU Enable/Disable. This bit should be set after programming all of the SPU configuration registers. 0 SPU is disabled 1 SPU is enabled
1 DIV	Div Option. Select Div4 and Div1 programmable options based on number of active sequences. 0 Use Div4 option, when multiple up to four sequences are active or only one sequence is active which does not require the highest resolution. 1 Use a Div1 option, when only one sequence (Sequence1) is active and highest resolution is required.
31:2	Reserved

NOTE

Div option mapping with active sequences must be correctly programmed by the user. No hardware checks are implemented.

72.5.6 Sample and hold mechanism

SPU supports Div4/Div1 sample and state processing frequency (both are effectively the same). For the Div1 option, the sample and state processing frequency is the SPU clock frequency. Likewise, for the Div4 option, the sample and state processing frequency is 1/4th the SPU clock frequency.

The Div1 or Div4 is related to hold circuitry (among other circuits as well), but the sample circuitry is always Div1. Therefore, if Div4 is selected for hold circuitry, then four samples are taken and then latched into a hold circuitry (for the next four Div1 clocks, synchronized for all inputs).

This sample and hold mechanism is applicable for external signals only, but counters logic process differently. These external signals may be considered as selected conditions for any sequence, and for any of the states of any sequence. There is no difference based on state or sequence, as all inputs are treated the same regardless.

72.5.6.1 Functional limitations

If there are multiple true conditions in a Div4 period, only one condition is registered. Never are there any missed true conditions, except for extra true conditions in a Div4 period. Therefore, the user should select the resolution of SPU based on number of active sequences.

- If the signal changes faster than the sample and state frequency, then the SPU would probably only detect one event per the sample period (for example: If there are multiple true conditions in a Div4 period, only one condition is registered).
- If the signal changes faster than the sample and state frequency, then the SPU would only be able to process one event per state processing frequency.

72.5.6.2 Assumption

When using Div1, SPU counter operations are not available for state evaluation in the immediately next clock cycle. Therefore, special handling must be done in state evaluation during this delayed effect to assure that it does not create undesirable results. Refer to counter workaround description in SPU actions.

When using Div4, however, there are no delayed effects when using SPU counter operations. This is because subsequent state evaluations are done four clocks apart, instead of every clock for Div1 selection. Therefore, counter workaround description in SPU actions, becomes redundant with Div4 option.

72.5.7 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions. List of the actions is given in SPU actions. The action triggers are mapped to the events by programming the action registers. Each event can be programmed to trigger interrupts, watchpoints, trace signals, counter stop/reset/increment and many other general actions. The following table shows the bit mapping for all the possible actions. All possible actions are divided in three groups: client based, counter based, system related.

Table 72-18. SPU action decoder

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
00	Client group	00000	CPU0	0000	No action	—	0=Stop 1=Start
		00001	CPU1	0001	DTM/gtm_gen0 trace	Valid for Cores, AHB, GTM, and trace group actions.	
		00010	CPU2	0010	OTM/gtm_gen1 trace		
		00011–00111	Reserved	0011	PTM/gtm_gen2 trace		
		01000	GTM	0100	WTM/gtm_gen3 trace		
		01001–01111	Reserved	0101	PMC1/NAR_supp_trig1	Valid for the cores as PMC; Valid for NAR as the suppress trigger actions	For PMC1/2/3/4 0= Stop 1= Start; For NAR suppress trigger = Don't care
		10000	NXMC0	0110	PMC2/NAR_supp_trig2		
		10001	NXMC1	0111	PMC3/NAR_supp_trig3		
		10010–10111	Reserved	1000	PMC4/NAR_supp_trig4		
		11000	NAR	1001	GTM_dtm_tr0	Valid for GTM	0=Stop 1=Start
		11001–11110	Reserved	1010	GTM_dtm_tr1		
				1011	GTM_ftm_tr0		
				1100	GTM_ftm_tr1		
		11111	Trace group[1]	1101–1111	Reserved	Valid for CPU cores and AHB	
01	Counter group	00000	Counter0	0000	No action	—	Don't care
		00001	Counter1	0001	Start timer/Incr counter	Valid for all counters/timers	
		00010	Counter2	0010	Stop timer		
		…	…	0011	Reset counter		
		01111	Counter15	0100	Capture		

Table continues on the next page...

Table 72-18. SPU action decoder (continued)

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
		10000–11111	Reserved	0101–1111	Reserved		
10	System group	00000	System action	0000	No action	—	Don't care
		00001–11111	Reserved	0001	Halt1_dci_evti	—	
				0010	Halt2_dci_evti	—	
				0011	Timestamp0_NXMC0	Valid for the NXMC clients	
				0100	Timestamp1_NXMC0		
				0101	Timestamp0_NXMC1		
				0110	Timestamp1_NXMC1		
				0111	Evto1	—	
				1000	Evto2	—	
				1001	NAR_sync_trigg	Valid for NAR	
				1010	Core INTR	—	
				1011–1111	Reserved	—	—
11	Reserved						

1. The valid actions for Trace group are OTM trace, PTM trace, DTM trace or WTM trace. No other actions can be selected with Trace group. The group can be defined using the Trace group configuration registers.

72.5.7.1 State n true action 0 (SnTA0)

The SPU can be programmed to enable up to four different actions against a True condition of every state. All the possible actions and their encoding are listed in Table 72-18. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1.

The following table shows the format of the SnTA0 register, where n, the state number, is equal to 0–7.

Offset: 0x01 (State0) Access: User read/write
0x03 (State1)
0x05 (State2)
0x07 (State3)
0x09 (State4)
0x0B (State5)
0x0D (State6)
0x0F (State7)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION2												0	0	0	0	ACTION1											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnTA0 register fields are described in the following table.

Table 72-19. SnTA0 register field descriptions

Field	Description
0:11 ACTION1	Action 1 to be performed for State n true condition. The decoding of these bits is defined in Table 72-18
12:15	Reserved
16:27 ACTION2	Action 2 to be performed for State n true condition. The decoding of these bits is defined in Table 72-18
28:31	Reserved

72.5.7.2 State n true action 1 (SnTA1)

The SPU can be programmed to enable up to four different actions against a True condition of every state. All the possible actions and their encoding are listed in Table 72-18. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1. The following table shows the format of the SnTA1 register, where n, the state number, is equal to 0–7.

Offset: 0x02 (State0), 0x04 (State1), 0x06 (State2), 0x08 (State3), 0x0A (State4), 0x0C(State5), 0x0E (State6), 0x10 (State7)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnTA1 register fields are described in the following table.

Table 72-20. SnTA1 register field descriptions

Field	Description
0:11 ACTION3	Action 3 to be performed for State n true condition. The decoding of these bits is defined in Table 72-18
12:15	Reserved
16:27 ACTION4	Action 4 to be performed for State n true condition. The decoding of these bits is defined in Table 72-18
28:31	Reserved

72.5.7.3 State n false action 0 (SnFA0)

The SPU can be programmed to enable up to four different actions against a False condition of every state. All the possible actions and their encoding are listed in Table 72-18. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. The following table shows the format of the SnFA0 register, where n, the state number, is equal to 0–7.

Offset: 0x11 (State0), 0x13 (State1), 0x15 (State2), 0x17 (State3), 0x19 (State4), 0x1B (State5), 0x1D (State6), 0x1F (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION2												0	0	0	0	ACTION1											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnFA0 register fields are described in the following table.

Offset: 0x12 (State0), 0x14 (State1), 0x16 (State2), 0x18 (State3), 0x1A (State4), 0x1C(State5) — Access: User read/write

Table continues on the next page...

0x1E (State6)

0x20 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnFA0 register fields are described in the following table.

Table 72-21. SnFA0 register field descriptions

Field	Description
0:11 ACTION1	Action 1 to be performed for State n false condition. The decoding of these bits is defined in Table 72-18
12:15	Reserved
16:27 ACTION2	Action 2 to be performed for State n false condition. The decoding of these bits is defined in Table 72-18
28:31	Reserved

72.5.7.4 State n false action 1 (SnFA1)

The SPU can be programmed to enable up to four different actions against a False condition of every state. All the possible actions and their encoding are listed in Table 72-18. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. The following table shows the format of the SnFA1 register, where n, the state number, is equal to 0–7.

Offset: 0x12 (State0) Access: User read/write

0x14 (State1)

0x16 (State2)

0x18 (State3)

0x1A (State4)

0x1C(State5)

0x1E (State6)

0x20 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnFA1 register fields are described in the following table.

Table 72-22. SnFA1 register field descriptions

Field	Description
0:11 ACTION3	Action 3 to be performed for State n false condition. The decoding of these bits is defined in Table 72-18
12:15	Reserved
16:27 ACTION4	Action 4 to be performed for State n false condition. The decoding of these bits is defined in Table 72-18
28:31	Reserved

72.5.7.5 Trace group configuration registers (DTMGC, PTMGC, OTMGC and WTMGC)

There are four action trace groups and a configuration register for each trace group:

- DTM trace—clients are configured with the DTM trace group configuration (DTMGC) register
- PTM trace—clients are configured with the PTM trace group configuration (PTMGC) register
- OTM trace—clients are configured with the OTM trace group configuration (OTMGC) register
- WTM trace—clients are configured with the WTM trace group configuration (WTMGC) register

Each of these groups are specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

The following table shows the format of the trace group configuration registers.

Offset 0x21 (DTMGC) 0x22 (PTMGC) 0x23 (OTMGC) 0x24 (WTMGC) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AHB2	AHB1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2	CPU1	CPU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The trace group configuration register fields are described in the following table.

Table 72-23. Trace group configuration register field descriptions

Field	Description
0 CPU0	0 Disable trace for CPU0 1 Enable trace for CPU0
1 CPU1	0 Disable trace for CPU1 1 Enable trace for CPU1
2 CPU2	0 Disable trace for CPU2 1 Enable trace for CPU2
15:3	Reserved
16 AHB1	0 Disable trace for AHB1 1 Enable trace for AHB1
17 AHB2	0 Disable trace for AHB2 1 Enable trace for AHB2
31:18	Reserved

72.5.7.6 Interrupt status (INTS)

The status of the external interrupt request to the processor core through the JTAGM is registered in the Interrupt status (INTS) register. This status bit is cleared when the acknowledge is received from the JTAGM. The following table shows the format of the INTS register.

Offset: 0x25 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The INTS register fields are described in the following table.

Table 72-24. INTS register field descriptions

Field	Description
0 INT	1 Interrupt status bit is cleared from JTAGM
31:1	Reserved

72.5.8 Counters control n (CCTRLn) registers

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. The following table shows the format of the CCTRLn registers, where n, the counter number, is equal to 0–15.

Offset 0x26–0x35 (CCTRL0–15) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	SW_RESET	PRESCALING			0	COUNT_INCR	MODE	ENABLE
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCTRLn register fields are described in the following table.

Table 72-25. CCTRLn register field descriptions

Field	Description
0 ENABLE	Enable. 0 Counter disabled 1 Counter enabled
1 MODE	Mode Select. Select counter or timer operation. 0 Counter 1 Timer
2 COUNT_INCR	Count Increment. 0 Increment the counter based on the actions generated from the sequences 1 Increment the counter based on the PMC input. The mapping of the PMC inputs with the counters is shown in Figure 72-11.
3	Reserved. Read returns 0.
6:4	Prescaling. Divides the timer clock by a defined value.

Table continues on the next page...

Table 72-25. CCTRLn register field descriptions (continued)

Field	Description
PRESCALING	000 No scaling, divide by 1 001 Divide by 4 010 Divide by 16 011 Divide by 32 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 SW_RESET	Reset. Counter can be reset by software 0 Software reset is disabled 1 Software reset
8 Reserved	This bit is for internal use only and must be set to 0.
31:9	Reserved. Read returns 0.

72.5.9 Counters compare n (CCMPn) registers

A 32-bit comparator value can be programmed for each of the counters. This value is compared with the corresponding counter/timer to generate an event for the SPU. The following table shows the format of the CCMPn registers, where n, the counter number, is equal to 0–15.

Offset: 0x36–0x45 (CCMP0–15) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMPARE_VAL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCMPn register fields are described in the following table.

Table 72-26. CCMPn register field descriptions

Field	Description
31:0 COMPARE_VAL	Compare value. This value is compared with the corresponding counter/timer value to generate an event for the SPU.

72.5.10 Status registers

72.5.10.1 Counter compare status (CCOMS)

The counter compare status register shows the status of the comparisons for all the counters. The bits in this register can be cleared by writing a 1 to them. The following table shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in Table 72-27. Counter compare values should not be changed dynamically. Thus, these fields should be cleared (by writing 1) only during these conditions:

- Counter is in disabled state (ENABLE bit of CCTRLn set to 0).
- Software reset bit (SW_RESET) is set.

Offset 0x46 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCOMS register fields are described in the following table, where n is the counter number.

Table 72-27. CCOMS register field descriptions

Field	Description
CCn	Counter n Compare. 0 Counter status is not true 1 Counter status is true

72.5.10.2 Counter overflow status (COS)

Whenever a counter overflow occurs, the corresponding counter overflow status register bit is set. The bits in this register can be cleared by writing a 1 to them. The following table shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in Table 72-28.

Offset 0x47 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CO15	CO14	CO13	CO12	CO11	CO10	CO9	CO8	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The COS register fields are described in the following table, where n is the counter number.

Table 72-28. COS register field descriptions

Field	Description
COn	Counter n Overflow. 0 Counter did not overflow 1 Counter overflow occurred

72.5.10.3 Counter capture status (CCAPS)

Whenever a counter value is captured, the corresponding capture status register bit for that counter is set. If the counter capture value causes a FIFO overflow, and thus the captured value is lost, the status bit is not set. The bits in this register can be cleared by writing a 1 to them. The following table shows the format of the CCAPS register.

All of the CC fields in the CCAPS register have the same description as shown in Table 72-29.

Offset 0x48 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCAPS register fields are described in the following table, where n is the counter number.

Table 72-29. CCAPS register field descriptions

Field	Description
CCn	Counter n Captured. 0 Counter not captured 1 Counter captured

72.6 Functional description

This section discusses the functional description of the SPU.

72.6.1 Input Mux unit

The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 16 inputs for each SLU. In addition to client input signals, there are certain performance events which are fed directly to the different counters inside the SPU.

The selection of 16 inputs for each state logic is done at two level of muxing both of which are shown in Figure 72-4.

- Level 1 Muxing—At this level, there are 209 inputs. Out of these possible inputs, 63 inputs are selected. This selection is done using an 8:1 Mux. Therefore, sixty-four 8×1 muxes are required. The set of 63 watchpoints are passed to the second level of muxing. The selection mechanism is shown in Figure 72-4.
- Level 2 Muxing—At this level, 16 inputs are selected out of the 63 possible inputs for each SLU. These 63 inputs consist of multiple set of 8 watchpoints from different clients, processor exception signals, interrupt from clients, and event IN/OUT signals. In addition, certain events from counters are used as feedback in the present state logic of each state.

  At level two there are 16 banks of 64:1 muxes for the selection of 16 inputs for one state logic. The inputs to these Muxes are expected to be somewhat unique. In general, care should be take to choose signal assignments that allow for key use cases

to be achieved without encountering concurrency restrictions. Since there are eight such states, 16 × 8 banks of 64:1 mux are required. The selection mechanism is shown in the following figure.

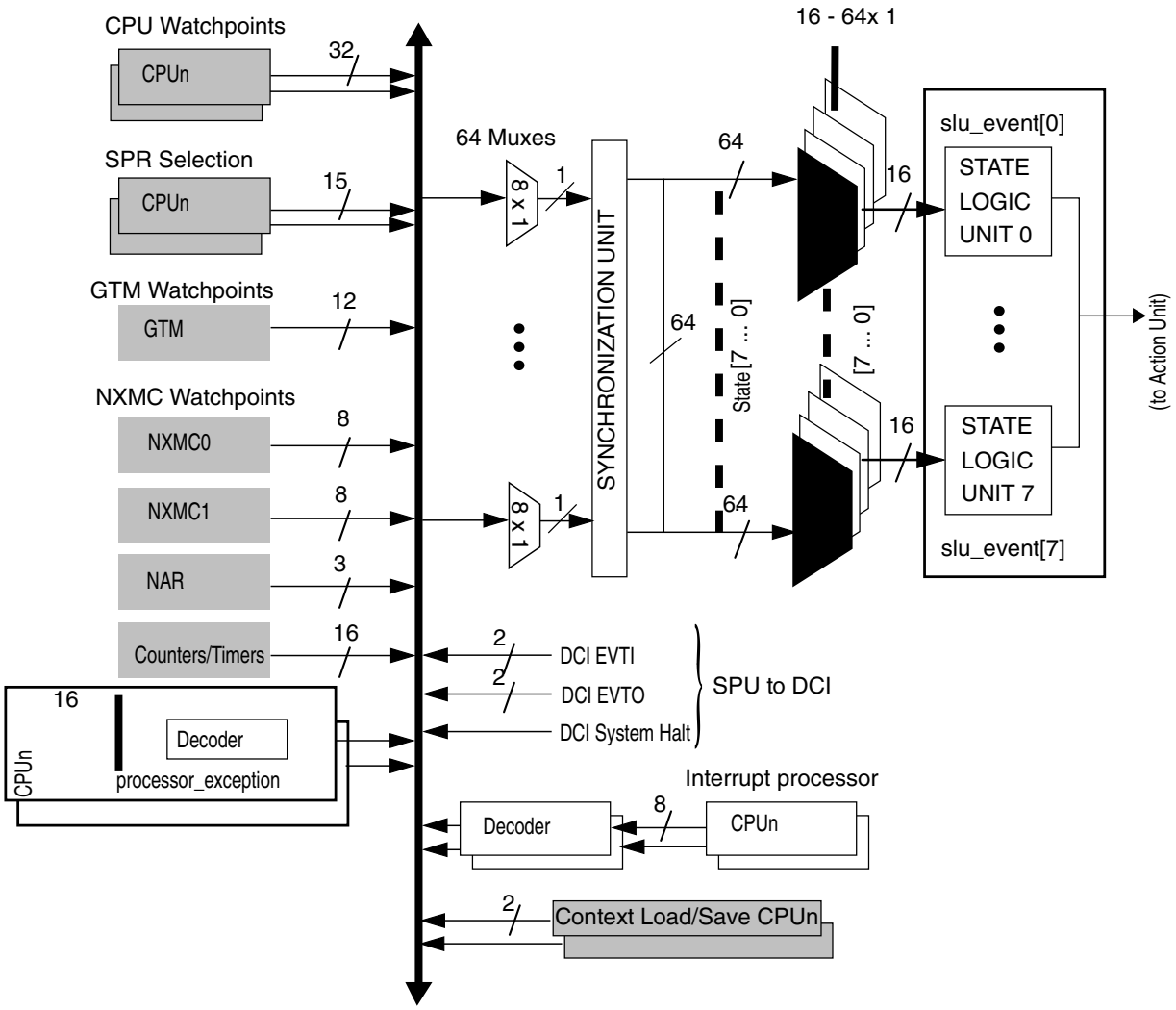


Figure 72-4. Input Mux unit block diagram

72.6.2 Processor exception vector encoding

Active-high input signals from the CPU indicate the particular exception being processed when the CPU exception enable signal is asserted. These signals can be used for debug triggering mechanisms. The SPU captures the CPU exception vector whenever the CPU exception enable signal is asserted and the SPU uses the exception vector as an input condition as shown in the following table. The CPU exception vector values from the CPUs must be byte swapped before they are programmed to the CPUn field of the CnPEVP registers as shown in the following table.

Table 72-30. Processor exception processing conditions

CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input—autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input—autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	0x6100	Program illegal
0x0062	0x6200	Program privileged
0x0070	0x7000	Performance monitor
0x0080	0x8000	System call
0x0090	0x9000	Debug
0x00A0	0xA000	EFPU data exception
0x00B0	0xB000	EFPU round exception
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	External input—vectored
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	Critical input—vectored

72.6.3 State logic unit (SLU)

Complex triggers and system performance monitor functions are implemented in the SLU. The SPU creates debug events based upon states in a sequence. A sequence can consist of a single state, or any number of states up to 8. Each state consists of combinational logic allowing AND/OR operations on inputs from the trigger source unit. Single or multiple (up to 8) actions can be triggered by a state machine.

A logic diagram of one SLU state is shown in the following figure.

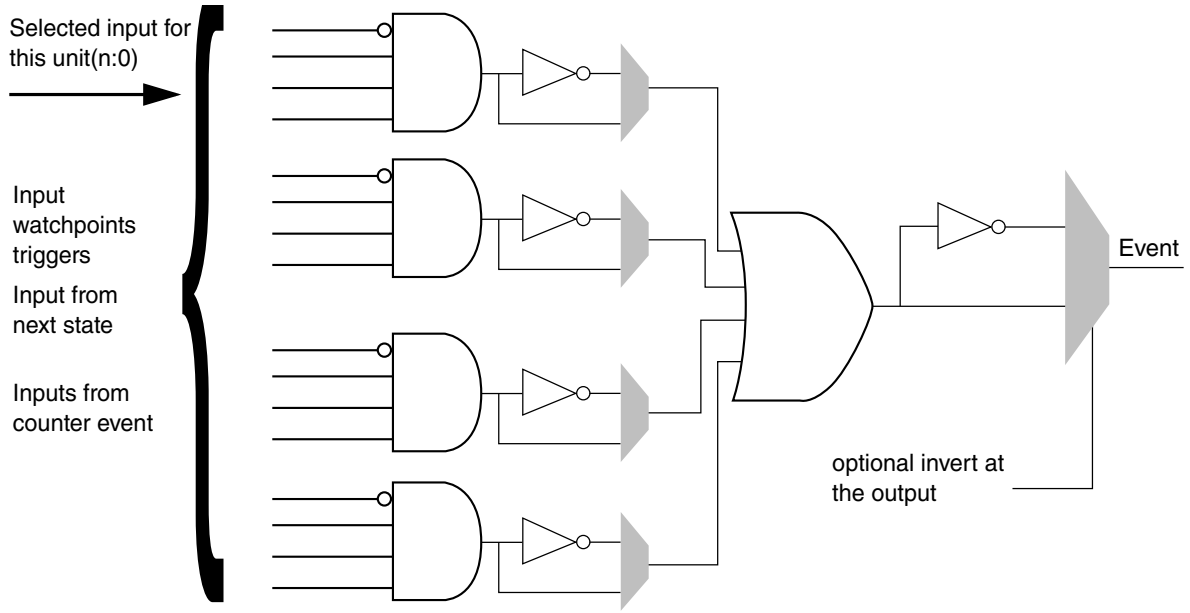


Figure 72-5. SLU block diagram

Each state consists of AND gates, the outputs of which are ORed together.

- Each AND gate can have up to 4 inputs, selected from the trigger source unit by the user.
- The user has an option to invert one input signal into the AND gate.
- The OR gate has a user selectable option to invert the output.
- The output of the OR gate is used as the trigger to move to the next state in the sequence.

The SLUs are responsible for performing operations on the selected input events to produce derived events. Each SLU has a control register that manages how the various inputs are combined to produce a derived event.

An event is triggered from each state based on all the possible selected inputs for each state. This event is used to trigger an Action.

Note

With only one optional inversion at the input of each AND gate and optional inversion at the output of the OR gate, many combinations are not possible.

72.6.4 Sequence formation

Event sequencing can be accomplished with the if-then-else operations. The next event can be fed back as a necessary condition for the current event. For example, event5 can be designated as a necessary condition for event4 in case test condition fails in event5. Event5 can be in turn be designated as necessary for event6 in case test condition passes, and so on. In this way, events can be chained together to create a sequence.

To create complex triggers, a configurable state machine is implemented. This allows the user to join states together with if-then-else operations to create a sequence. The SPU supports up to four simultaneous sequences, however only eight states are supported regardless of the number of active sequences, and each state can only be used in one unique sequence.

For example, the following state configurations are possible:

- 1 active sequence, with 8 states
- 2 active sequences, each with 4 states
- 2 active sequences, the first with 3 states and the second with 5 states
- 4 active sequences, each with 2 states
- 4 active sequences, each with 1 state

Each sequence has the following options which are shown in the following figure:

- The ability to optionally trigger one or more actions based on a true condition from any state in the sequence.
- The ability to optionally trigger one or more actions based on a false condition from any state in the sequence.
- Each state in a sequence has the ability to route to another state on a true condition from the state logic, and the ability to route to a different state based on a false condition from the state logic

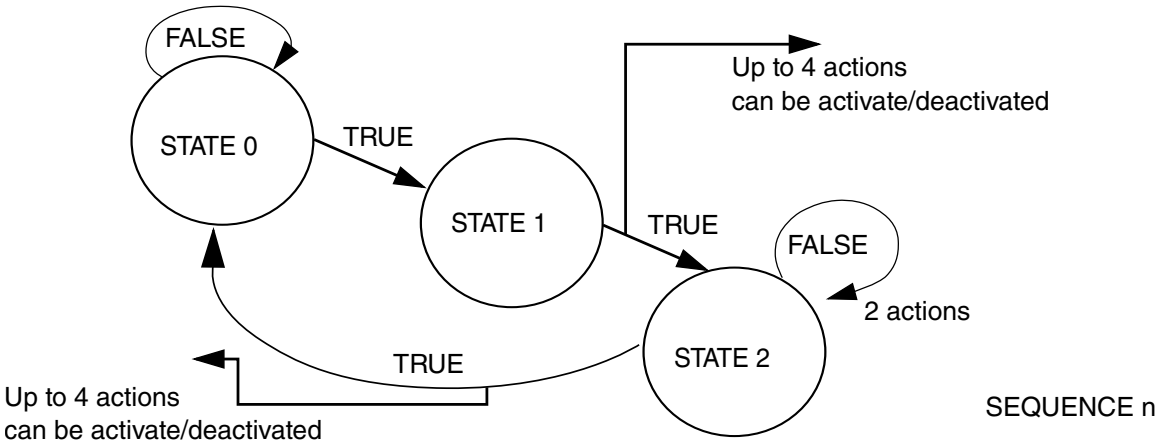


Figure 72-6. Example of a simple configurable if-then-else sequence

72.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in the following figure.

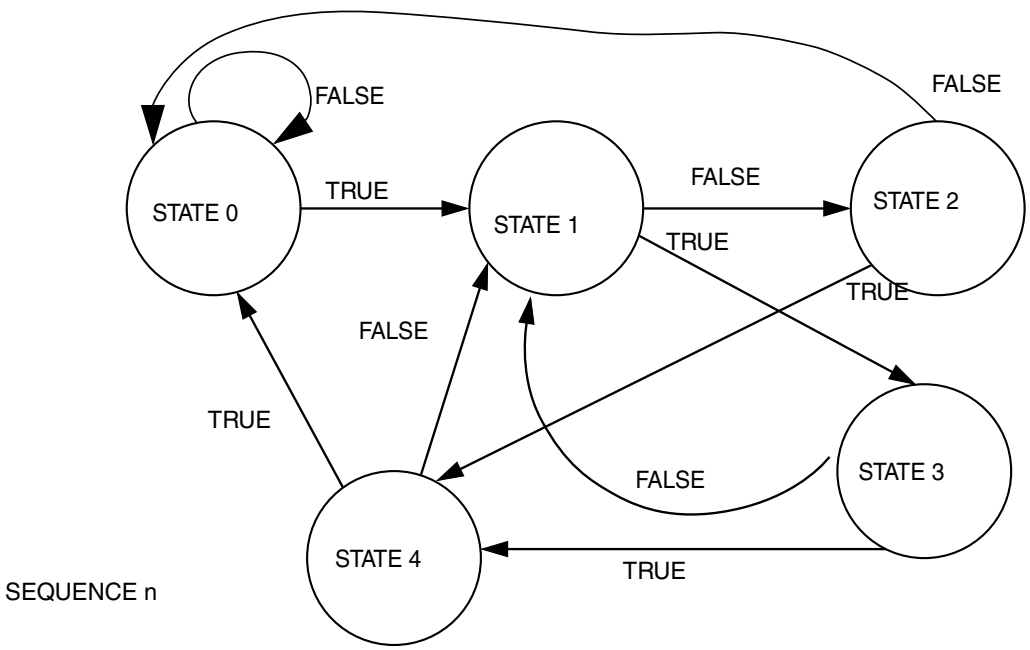


Figure 72-7. Complex sequence example

72.6.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered, and upon the 12th time of entering the function, stop the trace signal. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
  - If the value of counter[0] is less than 12, then the sequence moves back to State0.
  - If the value of couner[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

72.6.4.3 Sequence status register description

A status register providing feedback on each sequence is provided to allow debug tools to identify the source of a debug event from the SPU. The status register consists of two field types:

- States—each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register.
- Sequence status—each sequence has a field indicating the active state of the state machine. This is a read only field.

72.6.5 Performance counters/timers unit

The SPU counters are 32-bit counters that may be used to delay events, count distances, count latencies or count events as shown in the following figure. Therefore, these counter are used as either counters or timers. The selection between counter or timer for each of the counters is done via configuration register. The basic counting operation is straightforward. Once enabled (default is disabled mode), whenever the selected input is driven to a logic 1, the counter increments using the platform clock.

Each of the counter/timer registers are accessible (via the JTAG interface) for the configuration of timer/counter, the compare registers and global SPU timer/counter status registers. These register are accessible via the JTAG interface even when the SPU is in disabled mode.

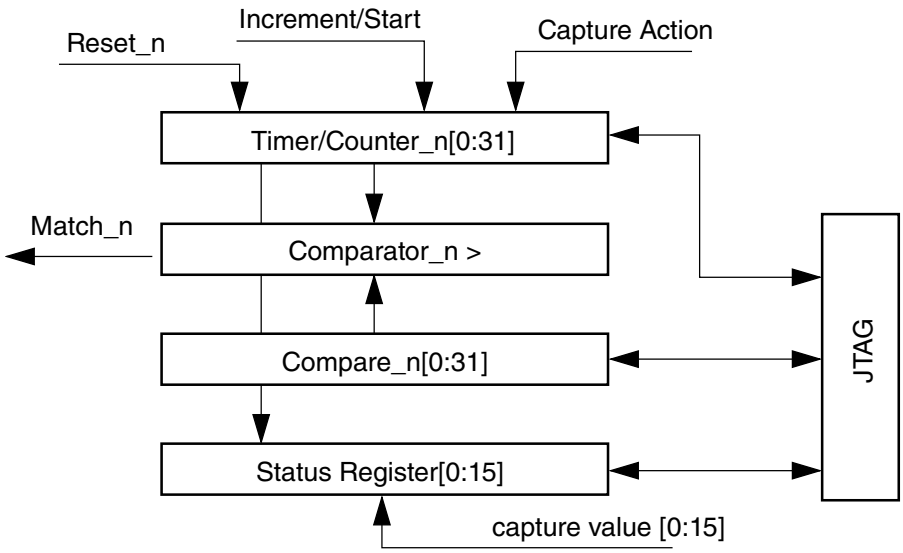


Figure 72-8. Counters/timers overview

72.6.5.1 Counter events and actions

Each counter is capable of detecting one event condition, such as compare value match. The compare value match is used for triggering actions in the SPU. Each counter can be reset in response to a reset action generated from the SPU. Capture registers are available for capturing the counter state in response to a SPU trigger condition. This capture information is updated in the status register which is accessible to users via JTAG interface. These status registers are if type write 1 to clear (W1C).

There are two classes of signals, actions and events:

- Actions—each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

  On an action to insert the timer/counter value into the trace stream, the value stored in the timer/counter is buffered inside the SPU and routed to the appropriate SoC block to insert the value into the trace stream one by one.

- Events—on a match event, whereby the a timer/counter value matches the value of the respective compare register, a signal is sent to the trigger source unit to indicate this.

72.6.5.2 Counter input selection at SPU level

Counter increment inputs derive from the SPU action or direct from the CPU performance measurement counters (CPU_PMC), as shown in the following figure. These inputs include signals from the primary SPU muxes or SPU events of counters.

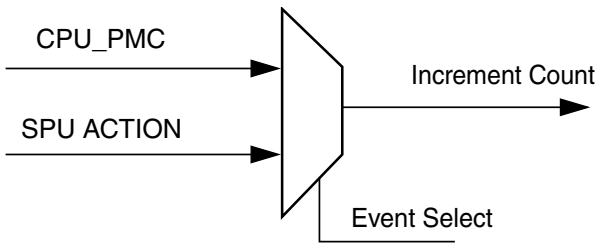


Figure 72-9. Input selection for count increment

72.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can optionally be routed from the SPU action unit, or directly from a CPU performance counter signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in the following figure.

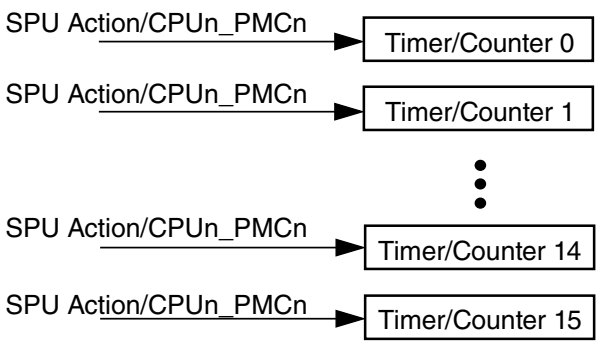


Figure 72-10. SPU counters increment assignment

72.6.5.4 CPU PMC mapping to counters

Each of the CPUs in the system has four performance measurement counters (PMC). The outputs (overflow signals) from these counters (of CPUn) are routed to the SPU debug unit. The counters inside the SPU can be configured to count the pulses from the PMCs. There is a one-to-one mapping between the PMC inputs and the counters that they trigger. Their mapping is shown in the following figure.

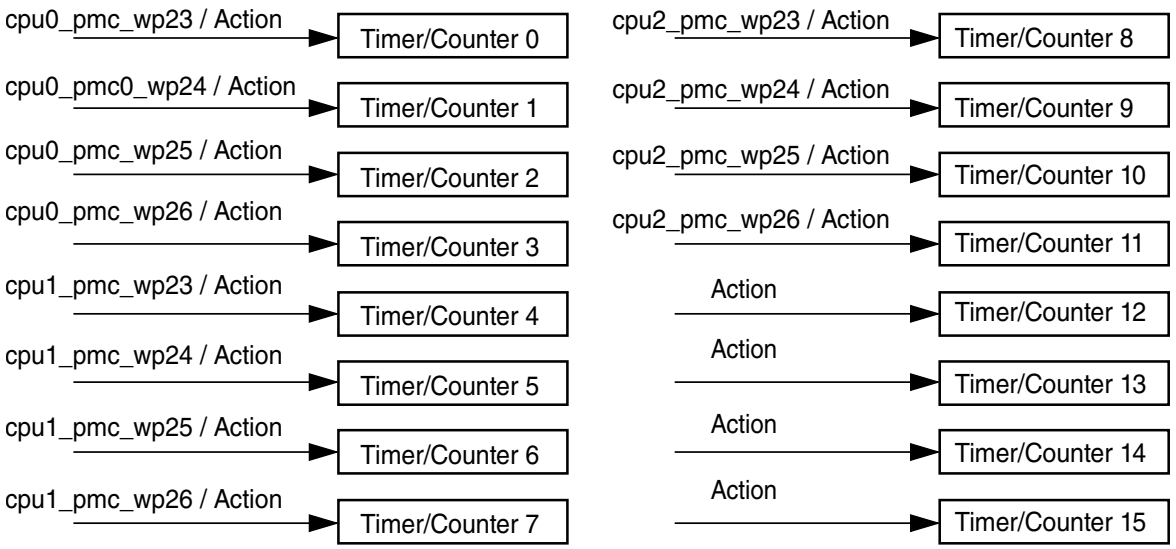


Figure 72-11. CPU PMC mapping inside the SPU unit

72.6.6 Action unit

The action unit block of the SPU is responsible for the generating the action triggers supported by various actions. A state has true and false (if-else-than) condition. Four simultaneous actions can be generated under true/false condition of a particular state. The list of the actions is given in the SPU actions. The actions triggers are mapped to the events by programming the set of SPU group action control for events registers. Each event can be programmed to trigger interrupts, timestamp watch points, trace signals, PMCs, start/reset/increment counters and many other general actions.

72.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers registers, defined in Table 72-23, are available where the user can group the different traces from same or different clients.

72.6.8 CKSRC and CKDATA format

The format of the CKSRC is defined for the SPU peripheral (as in-circuit trace client) as shown below:

ICT clients	CKSRC value
SPU counter trace client	b00_0000

Table continues on the next page...

ICT clients	CKSRC value
Reserved	b00_0001 — b11_1111

The format for the CKDATA field for the SPU counter trace client is as shown below:

CKDATA	
CKDATA[CTR_index]	CKDATA[CTR_value]
4 bits	32 bits

Chapter 73
JTAG Master (JTAGM)

73.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

73.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

- Configuration register
- Status register
- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

This figure shows the block diagram of the JTAGM.

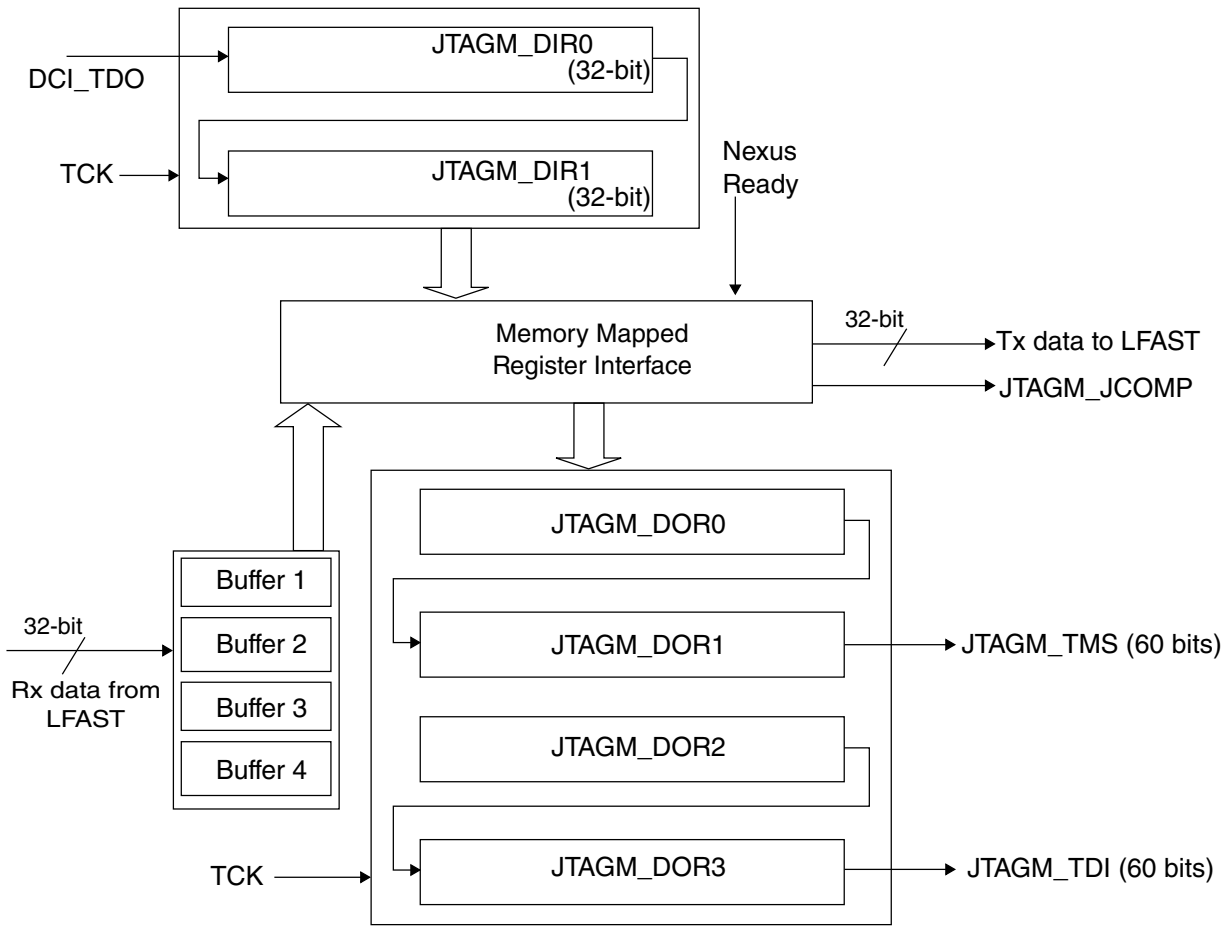


Figure 73-1. JTAGM block diagram

73.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

73.2 Functional description

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

This figure shows the flow diagram of the LFAST to DCI connection.

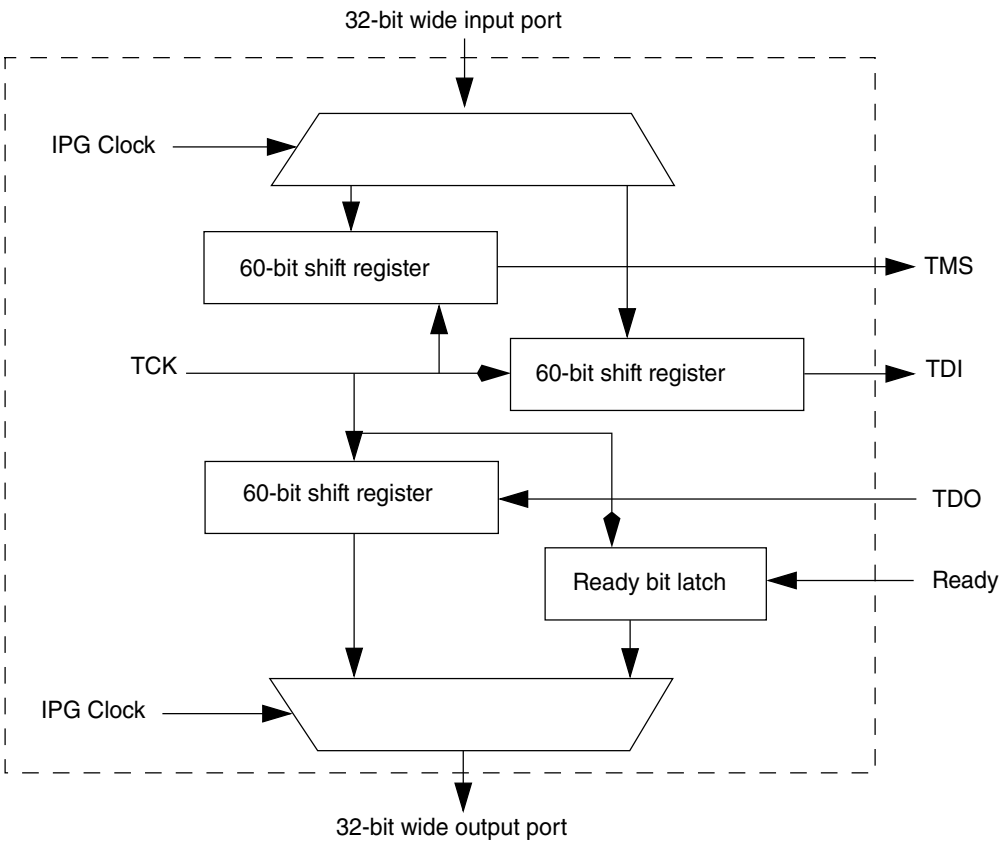


Figure 73-2. LFAST-to-DCI flow diagram

73.2.1 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAGM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
  - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
  - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

NOTE

The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

73.2.2 JTAGM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAGM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAGM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAGM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAGM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAGM status register (JTAGM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAGM to the LFAST also includes an 8-bit CRC. The data sent from the JTAGM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the

first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

73.2.3 JTAGM message tagging

The JTAGM, in conjunction with the LFAST, is pipelined to allow several JTAGM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAGM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAGM returns the response on channel E; if the tool sends a data command on channel F, the JTAGM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAGM are sent on channel A and response data is also sent back on channel A.

73.2.4 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

73.2.4.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

73.2.4.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

73.2.5 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus $\overline{\text{EVTI}}$ (Event In) and $\overline{\text{EVTO}}$ (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ signals to be embedded into the serial protocol. The $\overline{\text{EVTI}}$0 and $\overline{\text{EVTI}}$1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either $\overline{\text{EVTO}}$0 or $\overline{\text{EVTO}}$1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI.

The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

73.2.6 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in Status register ready bit). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written +
- 32-bit value to be written to the configuration register +
- 24-bit mask of which bits within the status register should be written +
- 8 bits of zero-padding +
- 24-bits of data to be written to the status register +
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data).

The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module.

The JTAGM requests data from the LFAST only when at least one internal buffer is free.

73.2.7 JTAGM to DCI serial interface

This figure shows the data transfer timings between JTAGM and DCI.

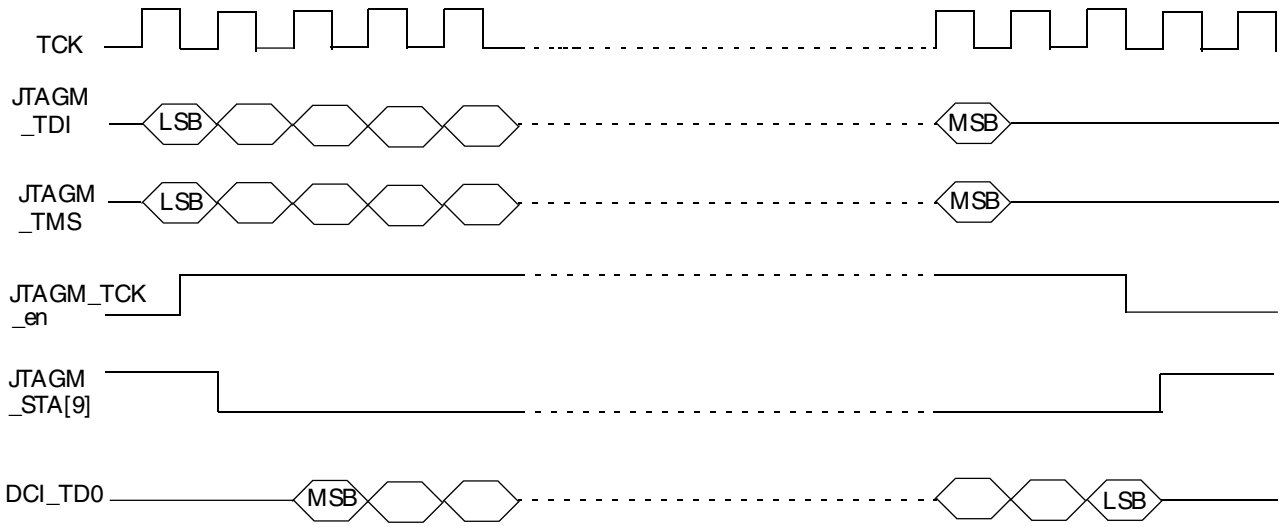


Figure 73-3. JTAGM to DCI serial interface diagram

73.2.8 JTAGM data handling and CRC calculation in LFAST mode

This figure shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

bit 127 to bit 68		bit 67 to bit 8		7	6	5	4	3	2	1	0
TMS (60 bits)		TDI (60 bits)		CRC (8 bits)							
bit 0 to bit 59		bit 0 to bit 59									
Last TMS bit transmitted by JTAGM	First TMS bit transmitted by JTAGM	Last TDI bit transmitted by JTAGM	First TDI bit transmitted by JTAGM								

Figure 73-4. Bit ordering for LFAST to JTAGM transfer

This figure shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

bit 95 to bit 36		bit 35 to bit 32	bit 31 to bit 8	7	6	5	4	3	2	1	0
TDO (60 bits)		0 pad (4 bits)	JTAGM_SR (24 bits)	CRC (8 bits)							
bit 0 to bit 59		bit 0 to bit 59	bit 31 to bit 8								
First TDO bit received by JTAGM	Last TDO bit received by JTAGM										

Figure 73-5. Bit ordering for JTAGM to LFAST transfer

The CRC calculation is described in following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in Figure 73-5).

73.2.9 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

73.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

73.4 Memory mapped registers

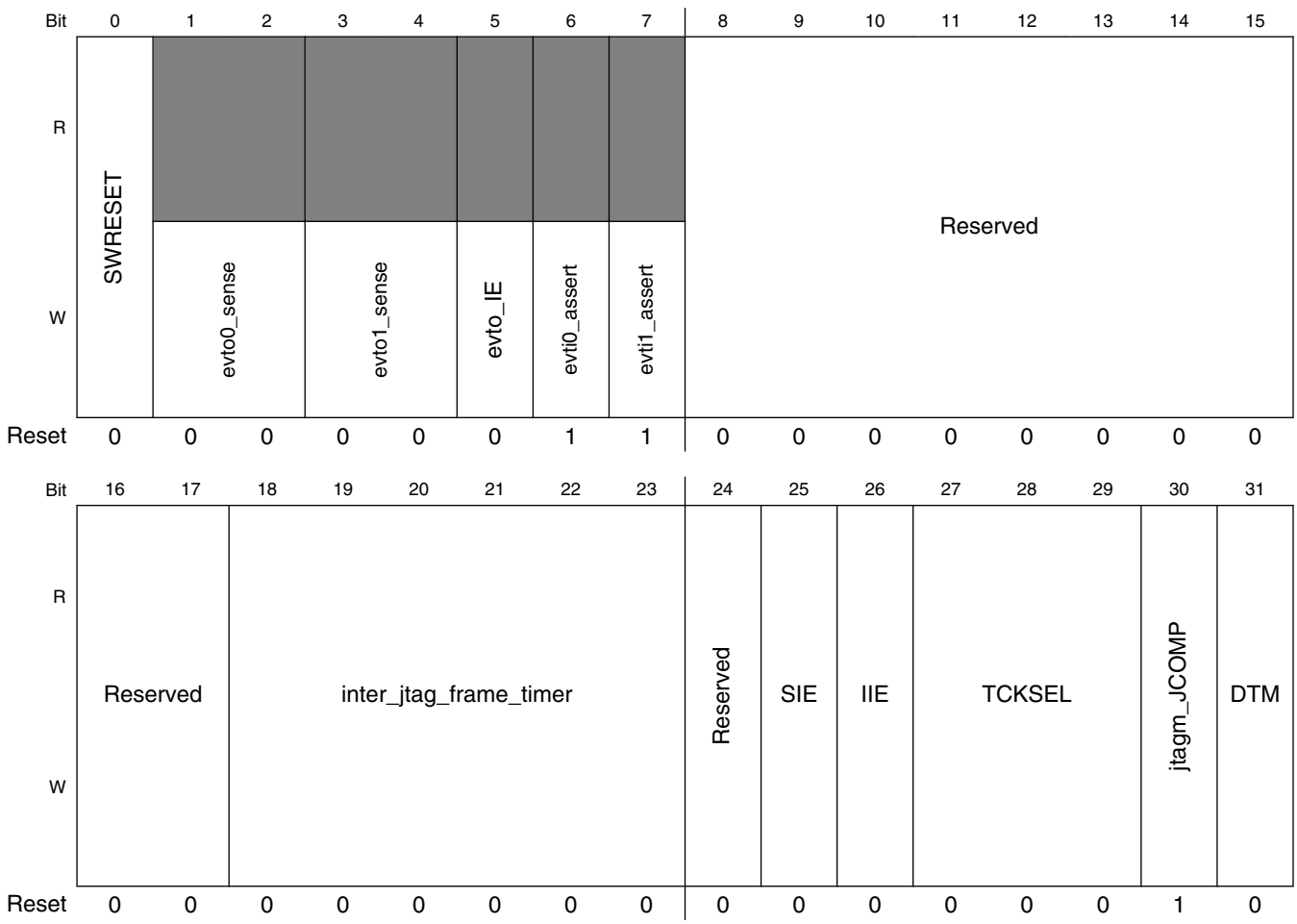
JTAGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JTAGM_MCR)	32	R/W	0300_0002h	73.4.1/3955
4	Status Register (JTAGM_SR)	32	R/W	0020_0200h	73.4.2/3958
8	Data Out Register 0 (JTAGM_DOR0)	32	R/W	0000_0000h	73.4.3/3960
C	Data Out Register 1 (JTAGM_DOR1)	32	R/W	0000_0000h	73.4.4/3960
10	Data Out Register 2 (JTAGM_DOR2)	32	R/W	0000_0000h	73.4.5/3961
14	Data Out Register 3 (JTAGM_DOR3)	32	R/W	0000_0000h	73.4.6/3961
18	Receive CRC Register (JTAGM_RxCRC)	32	R	0000_0000h	73.4.7/3962
1C	Data Input Register 0 (JTAGM_DIR0)	32	R	0000_0000h	73.4.8/3962
20	Data Input Register 1 (JTAGM_DIR1)	32	R	0000_0000h	73.4.9/3962

73.4.1 Module Configuration Register (JTAGM_MCR)

This register contains the status bits for the current transfer.

Address: 0h base + 0h offset = 0h



JTAGM_MCR field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit; after being written with a 1, this bit is auto-cleared within 3 TCK cycles. Writing a 0 has no effect.
1–2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO0}}$ 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$ 0 edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$0 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$0 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$0

Table continues on the next page...

JTAGM_MCR field descriptions (continued)

Field	Description
3–4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$ 1 line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$ 1 edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$ 1 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$ 1 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$ 1
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt. 0 No JTAGM interrupt is generated In response to any $\overline{\text{EVTO}}$ 0 or $\overline{\text{EVTO}}$ 1 activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	$\overline{\text{EVTI}}$0 Assert. 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	$\overline{\text{EVTI}}$1 Assert. In software mode (DTM = 1), the internal $\overline{\text{EVTI}}$ signal remains de-asserted, regardless of the state of this bit and writing to this bit has no effect. 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted
8–17 Reserved	This field is reserved.
18–23 inter_jtag_frame_timer	TCK delay. 000000 0 TCK delay 000001 1 TCK delay ... ... 111111 63 TCK delay
24 Reserved	This field is reserved.
25 SIE	SPU Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27–29 TCKSEL	TCK clock frequency selection. When the JTAGM is enabled, this bit should not be programmed with 000. 000 Reserved 001 TCK is system clock ÷ 2 010 TCK is system clock ÷ 3 011 TCK is system clock ÷ 4 100 TCK is system clock ÷ 5 101 TCK is system clock ÷ 6 110 TCK is system clock ÷ 7 111 TCK is system clock ÷ 8

Table continues on the next page...

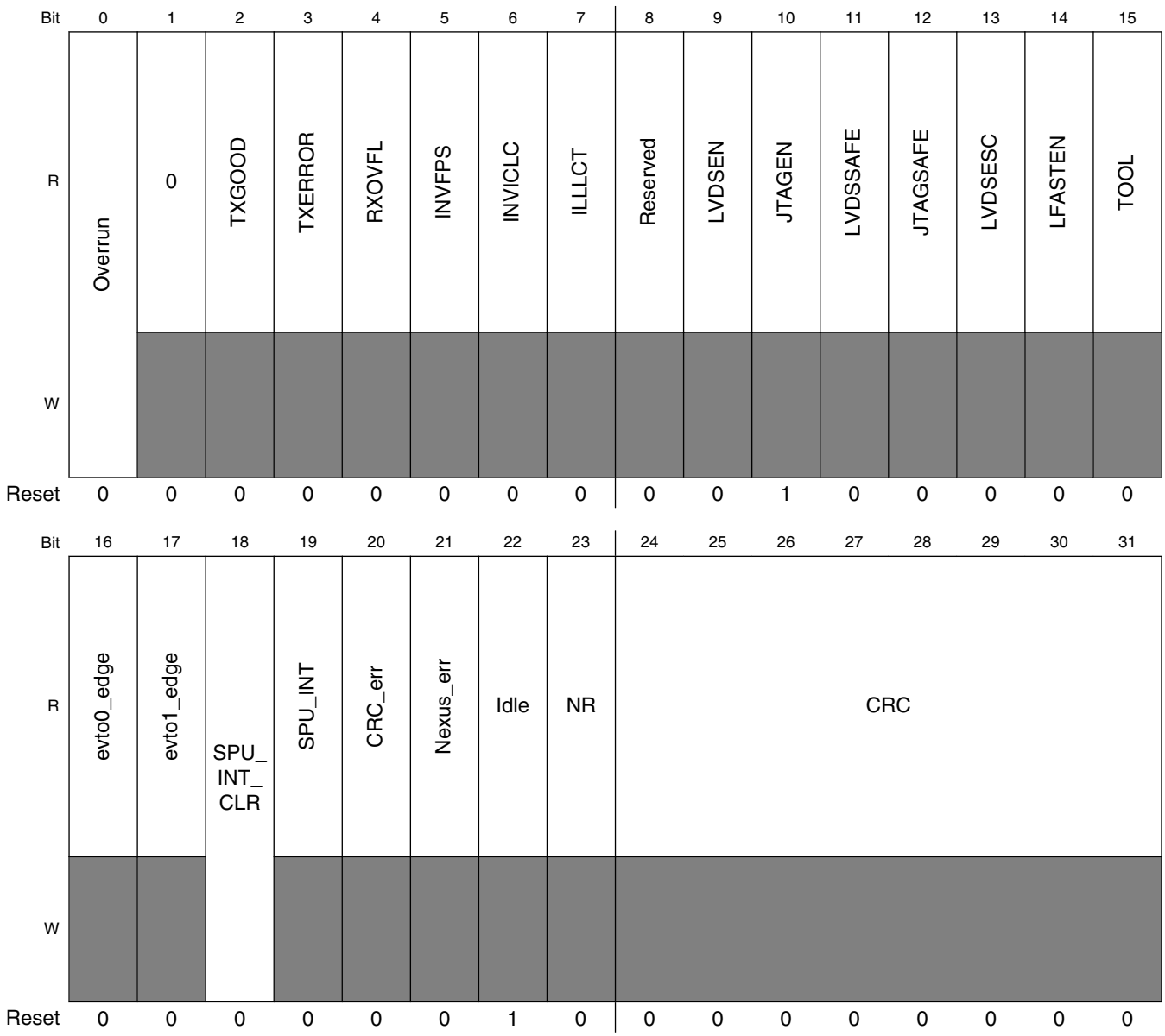
JTAGM_MCR field descriptions (continued)

Field	Description
30 jtagm_JCOMP	JTAG reset. 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

73.4.2 Status Register (JTAGM_SR)

The functionality of some of the SR bits changes depending on the setting of the DTM bit in the MCR. These differences are described in the field descriptions table.

Address: 0h base + 4h offset = 4h



JTAGM_SR field descriptions

Field	Description
0 Overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 TXGOOD	Status information from the LFAST to be used by software: Indicates either data frame or ping frame was transmitted successfully
3 TXERROR	Status information from the LFAST to be used by software: Indicates Tx data Interface not enabled and a frame is ready to be transmitted
4 RXOVFL	Status information from the LFAST to be used by software: Indicates Rx data FIFO overflow
5 INVFPS	Status information from the LFAST to be used by software: Indicates reception of frame with invalid frame payload size
6 INVICLC	Status information from the LFAST to be used by software: Indicates reception of frame with invalid ICLC code
7 ILLLCT	Status information from the LFAST to be used by software: Indicates reception of illegal LCT frame
8 Reserved	This field is reserved. Reserved to 0
9 LVDSEN	DCI status: DCI LVDS port enable
10 JTAGEN	DCI status: DCI JTAG port enable
11 LVDSSAFE	DCI status: DCI LVDS safe mode
12 JTAGSAFE	DCI status: DCI JTAG safe mode
13 LVDSESC	DCI status: DCI LVDS escape mode
14 LFASTEN	DCI status: DCI enable LFAST
15 TOOL	DCI status: DCI tool present, set when a debug tool is connected
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit. 0 No activity sensed on $\overline{\text{EVTO}}$ 0 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO}}$ 0
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO}}$ 1 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO}}$ 1
18 SPU_INT_CLR	Write 1 clears the SPU_INT. This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit.

Table continues on the next page...

JTAGM_SR field descriptions (continued)

Field	Description
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to 1 when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. In LFAST mode (DTM = 0), this bit has no effect and should not be used. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read
24–31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode (DTM = 0): Bit 24:31 contains the CRC. Write has no effect. SW Mode (DTM = 1): • Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. • Writing 1 on bit 24 clears evto0_edge, bit 16 • Writing 1 on bit 25 clears evto1_edge, bit 17

73.4.3 Data Out Register 0 (JTAGM_DOR0)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TMS_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR0 field descriptions

Field	Description
0–31 TMS_HIGH	Higher word of data for TMS, bits 59-28. Writable by software only when DTM = 1

73.4.4 Data Out Register 1 (JTAGM_DOR1)

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TMS_LOW																												Reserved			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR1 field descriptions

Field	Description
0–27 TMS_LOW	Lower word of data for TMS, bits 27-0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first
28–31 Reserved	This field is reserved.

73.4.5 Data Out Register 2 (JTAGM_DOR2)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDI_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR2 field descriptions

Field	Description
0–31 TDI_HIGH	Higher word of data for TDI, bits 59-28. Writable by software only when DTM = 1

73.4.6 Data Out Register 3 (JTAGM_DOR3)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TDI_LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

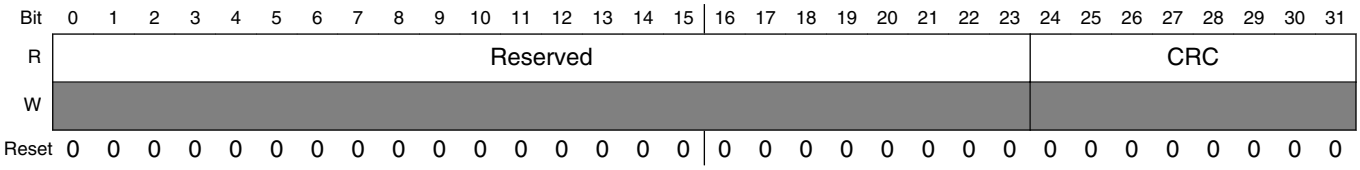
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDI_LOW												Reserved			
W																Send
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR3 field descriptions

Field	Description
0–27 TDI_LOW	Lower word of data for TDI, bits 27-0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
28–30 Reserved	This field is reserved.
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

73.4.7 Receive CRC Register (JTAGM_RxCRC)

Address: 0h base + 18h offset = 18h

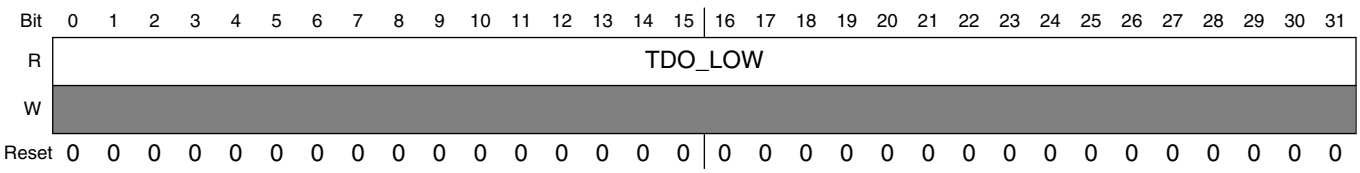


JTAGM_RxCRC field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

73.4.8 Data Input Register 0 (JTAGM_DIR0)

Address: 0h base + 1Ch offset = 1Ch

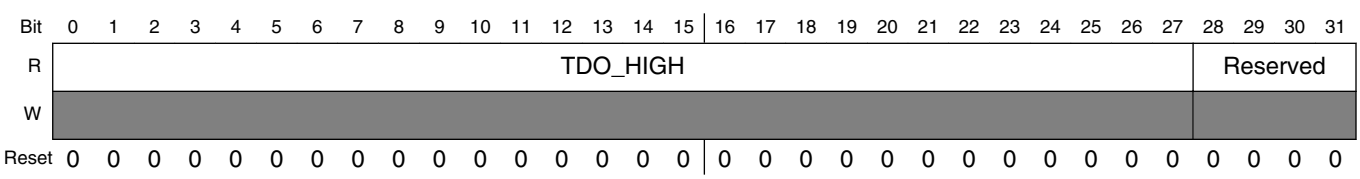


JTAGM_DIR0 field descriptions

Field	Description
0–31 TDO_LOW	Lower word of data received on TDO, bits 0-31

73.4.9 Data Input Register 1 (JTAGM_DIR1)

Address: 0h base + 20h offset = 20h



JTAGM_DIR1 field descriptions

Field	Description
0–27 TDO_HIGH	Higher word of data received on TDO, bits 32-59
28–31 Reserved	This field is reserved.

Chapter 74
Debug Zipwire

74.1 Overview

The Debug SIPI and Debug LFAST modules work together as a single unit called Debug Zipwire. The Debug LFAST portion of the two modules allows for high speed debugging capabilities.

The Debug LFAST operates in slave mode configuration. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

74.2 Debug Zipwire Overview

Dedicated Zipwire functionality is available for debug use allowing support for increased bandwidth read/write operations to the chip memory space performed over the debug interface. This Debug Zipwire function is available on both production and ED devices. The modules used to support Debug Zipwire are the Debug SIPI module, the Debug LFAST module and the LFAST Switch. The Debug Zipwire functionality does not use the LFAST or SIPI modules which support the Zipwire interprocessor bus. The only resource shared between the Debug and interprocessor bus Zipwire interfaces is the use of a single shared LFAST PLL on the Production Device (PD).

The Debug Zipwire interface allows a connected tool to perform 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address in the microcontroller. The Zipwire architecture is fully pipelined and supports multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock for a total of five pins. The Debug Zipwire interface does not support streaming mode commands.

For single random access, 32-bit read, from any 32-bit address location, the latency is approximately 1 µs.

74.3 Debug Zipwire Block Diagram

Figure 74-1 shows a tool connected to the MCU via five wires. These five pins are shared with the standard JTAG pins, but for simplicity this is not show on the diagram. The MCU Debug Zipwire only supports the Target mode of Zipwire. The Debug SIPI Bus Master Interface is used for all Target Mode transactions. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST Reference clock coming from the MCU clock system.

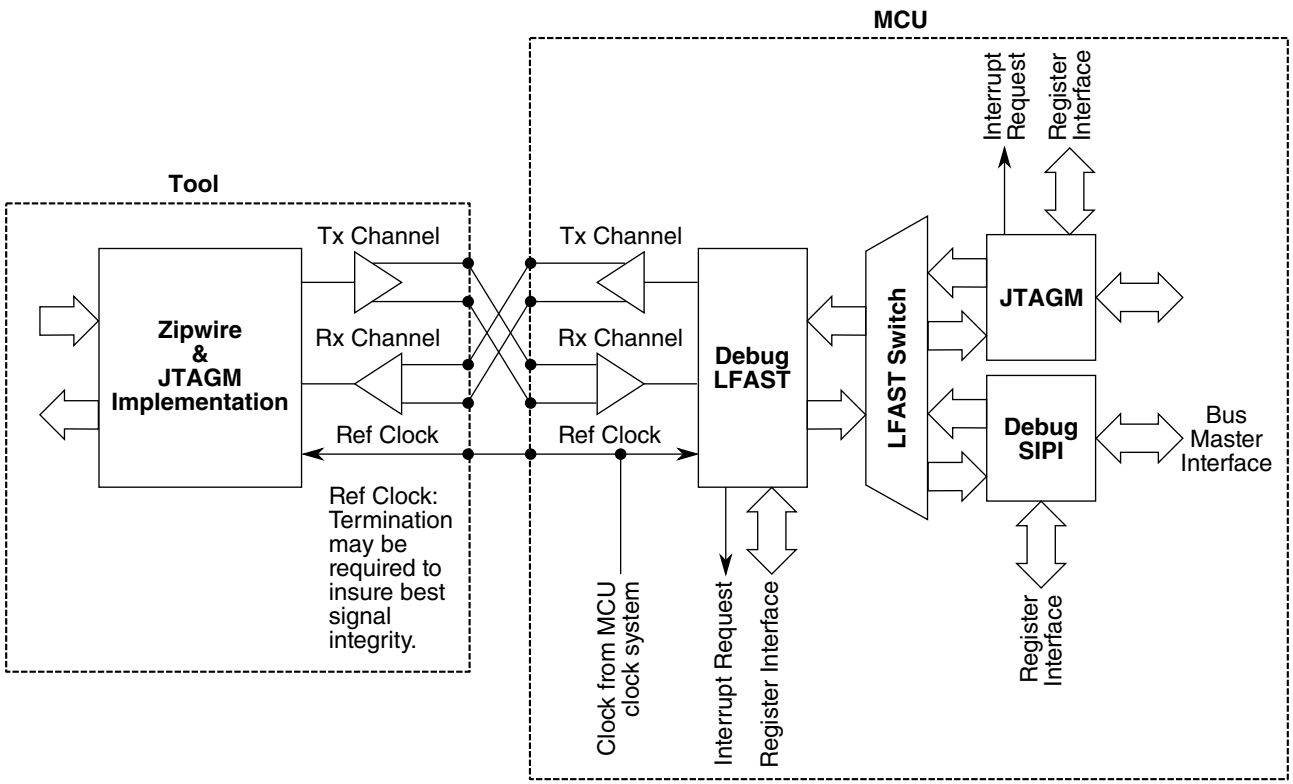


Figure 74-1. Debug Zipwire connection diagram

74.4 Architecture

Debug Zipwire is composed of several modules and system resources. The physical layer is LVDS with 1.2 V common mode voltage and 200mV swing. The pads are shared with JTAG signals.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.
- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). Debug SIPI runs on top of Debug LFAST and is fully encapsulated within the Debug LFAST payload. Debug SIPI uses three fixed sizes of Debug LFAST payload in Debug Zipwire:

- 32-bit
- 64-bit
- 96-bit

The SIPI protocol implements a suite of commands initiated by a tool, for reading and writing any 32-bit address location in a connected MCU. Only the tool can initiate commands. The MCU only responds to commands initiated by the tool and never creates any unsolicited messages.

For Debug Zipwire, a reduced version of the SIPI module called Debug SIPI is used. The Debug SIPI module implements only the recipient part of the SIPI protocol (called Target mode). The Debug SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the inititating tool, Debug SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

74.5 Debug Zipwire implementation

Figure 74-2 shows shows the modules on the production device which support the Debug Zipwire function, along with their interconnections to the debug interface signals.

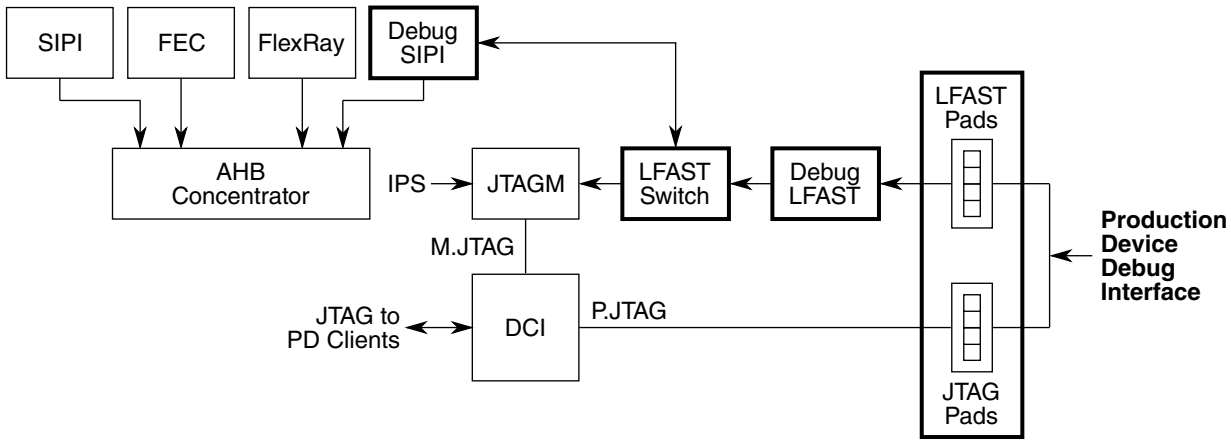


Figure 74-2. Block diagram showing connections for Production Device

Internal connections to other debug modules which are used to support other debug interface protocols such as JTAG and JTAG over LFAST are also shown. The Debug SIPI module is connected to the Debug LFAST module via an LFAST switch module, which controls routing of debug LFAST traffic to either the Debug SIPI or the JTAGM module.

Figure 74-3 shows the equivalent connections for ED devices utilizing the LFAST switch.

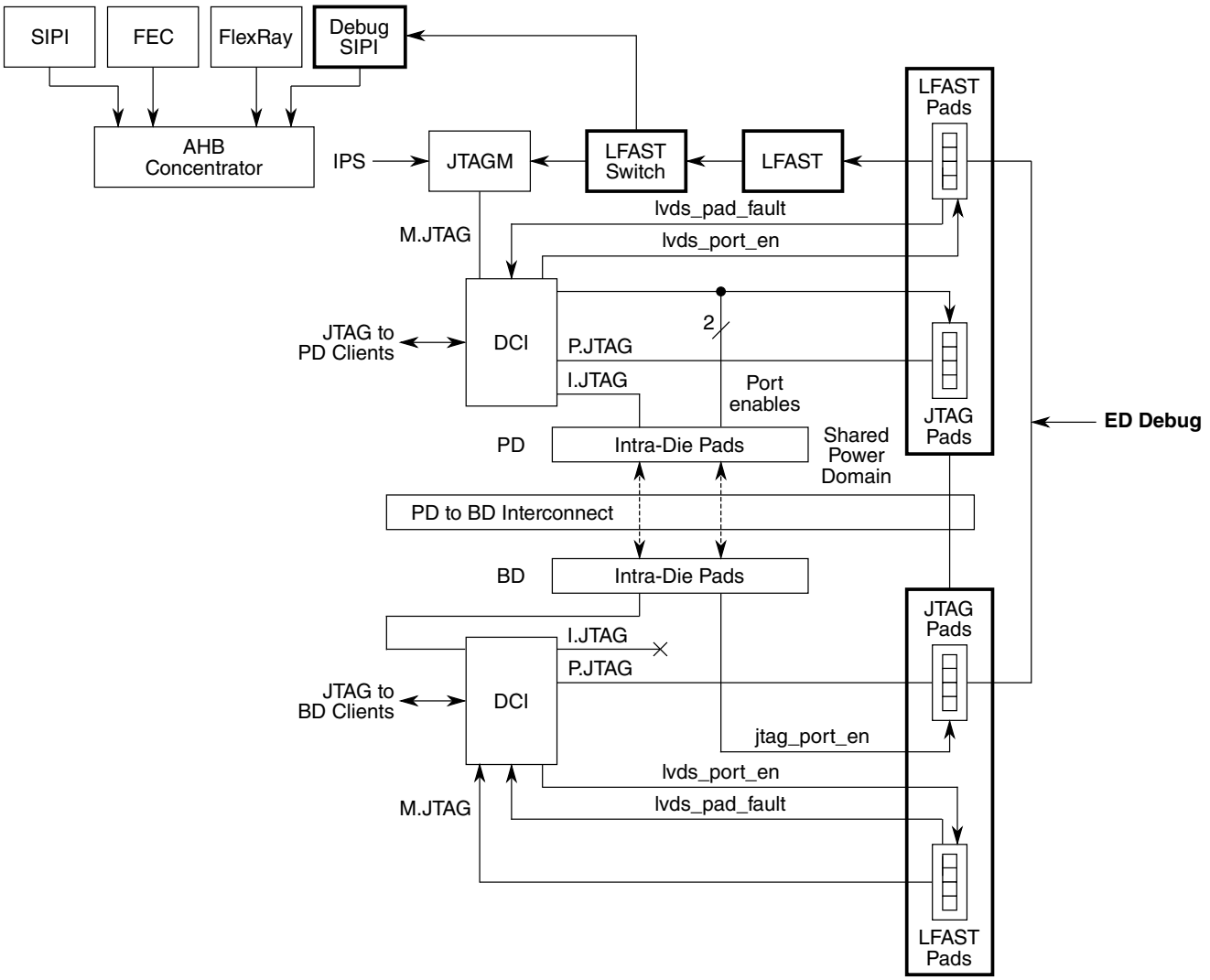


Figure 74-3. Block diagram showing connections for the ED Device

74.6 LFAST clocking

The LFAST implementation includes a dedicated PLL for multiplying a low speed clock source up to 320 MHz for the transmission and reception of data. This PLL is used to support both interprocessor bus Zipwire and Debug Zipwire so care must be taken to ensure that any tool configuration of this resource is compatible with application usage of Interprocessor Bus Zipwire. The high speed asynchronous nature of LFAST, requires that both the local node and remote node, operate from the same, stable, low jitter clock sources. This common low speed clock is referred to a the Reference Clock.

The MCU includes clock multiplexing to allow the source of the LFAST reference clock to come from one of three sources via a programmable clock divider:

- XOSC
- Output of PLL0 (PLL0:PHI)
- External pin

The most expedient source for the reference clock is the 20 MHz crystal oscillator (XOSC) from one MCU. This 20 MHz clock can be used as the LFAST Reference clock for the local LFAST. This same 20 MHz clock must be exported from the local MCU to the remote MCU, to be used as the LFAST reference at the remote MCU.

The 20 MHz clock may be used in the following ways:

- Shared between the MCUs using a dedicated reference clock pin on each MCU
- Divided by 2 and shared as a 10 MHz reference
- Shared as the fundamental clock for the MCU system clock and the LFAST reference clock

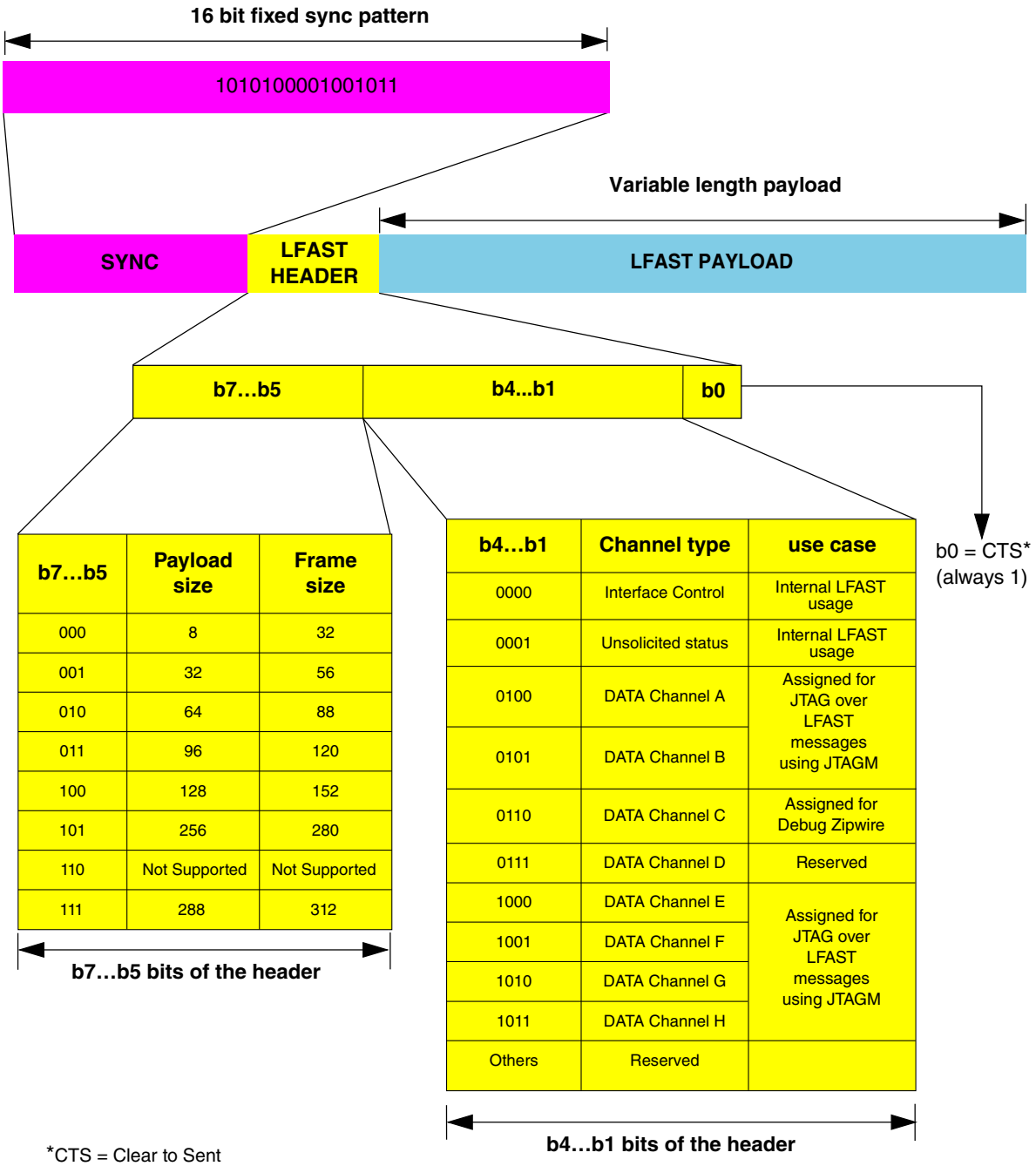
The LFAST PLL for generating the 320 MHz clock, is shared between LFAST for SIPI.

74.7 LFAST switch

Incoming messages on the Debug LFAST interface can be either for JTAG over LFAST use through the JTAGM or debug Zipwire messages. Differentiation of these messages is via use of LFAST channel number. DATA channel C is used to identify high speed Debug SIPI accesses, while JTAG over LFAST messages are supported using LFAST channels A, B, E, F, G, H as shown in Figure 74-4

The LFAST SWITCH module is used to direct LFAST messages based on the above scheme, with JTAG over LFAST messages being routed to the JTAGM IP block and debug Zipwire messages being routed to the Debug SIPI block.

The LFAST switch has no accessible registers and requires no user configuration.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	Not Supported	Not Supported
111	288	312

b4…b1	Channel type	use case
0000	Interface Control	Internal LFAST usage
0001	Unsolicited status	Internal LFAST usage
0100	DATA Channel A	Assigned for JTAG over LFAST messages using JTAGM
0101	DATA Channel B	
0110	DATA Channel C	Assigned for Debug Zipwire
0111	DATA Channel D	Reserved
1000	DATA Channel E	Assigned for JTAG over LFAST messages using JTAGM
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 74-4. LFAST channel allocation

74.8 Debug SIPI module

The debug SIPI module is used to support high bandwidth read/write accesses over the LFAST debug interface. It is dedicated for debug use and is entirely separate from the SIPI module used to support interprocessor bus zipwire interface.

The debug SIPI characteristics are shown in Table 74-1.

Table 74-1. Debug SIPI feature set

Feature	Debug implementation
Number of channels supported	1
Initiator support	No
Target support	Yes
Block transfer support	No
Chip XBAR connection	M1 on XBAR_1

The debug SIPI module issues read and write accesses via the AHB concentrator connected to physical master port 2 on XBAR_1, which is shared with the ethernet, FlexRay and Zipwire SIPI bus masters.

The debug SIPI memory mapped registers are accessible via PBRIDGE_0 (off platform - slot 10) with base address FFFD_4000h.

The debug SIPI has Logical Bus Master Assignment of 6 for SMPU0 and SMPU1, an NXMC trace source ID (SRC) of 1101b and a trace master ID (MASTER) = 1011b.

The debug SIPI supports only target mode SIPI operations on a single channel with a reduced command set. Commands supported include read (8/16/32 bits) and write (8/16/32 bits) along with the appropriate responses. Other SIPI commands are unimplemented as noted in Table 74-2.

The debug SIPI will be capable of:

- receiving a 64-bit command on LFAST channel C and responding with either a 64-bit response or 32-bit error response on LFAST channel C.
- receiving a 96-bit command on LFAST channel C and responding with either a 32-bit acknowledge or 32-bit error response on LFAST channel C.

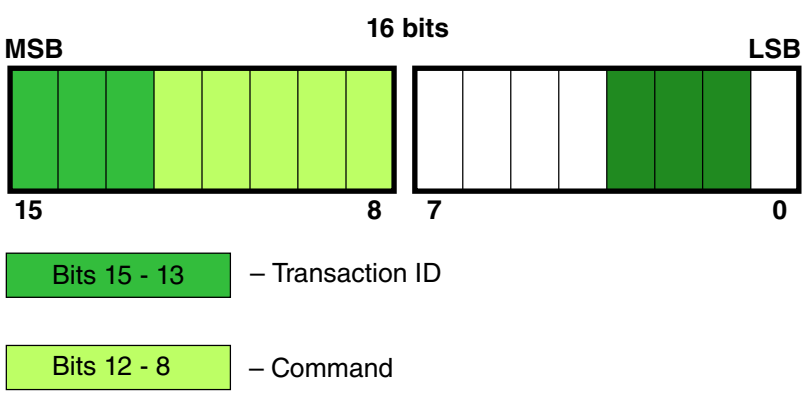


Figure 74-5. Debug SIPI header

NOTE

See Figure 74-4 for description of bits[7:0]

Table 74-2. SIPI header command coding

Bits[12:8] (hex)	Command	Payload Size	Message Direction at chip (SIPI Target)
00	Read 8 bits (Received by SIPI from LFAST)	64	Rx
01	Read 16 bits (Received by SIPI from LFAST)	64	Rx
02	Read 32 Bits (Received by SIPI from LFAST)	64	Rx
03	Reserved	—	—
04	Write 8 bits with ACK (Received by SIPI from LFAST)	96	Rx
05	Write 16 bits with ACK (Received by SIPI from LFAST)	96	Rx
06	Write 32 bits with ACK (Received by SIPI from LFAST)	96	Rx
07	Reserved	—	—
08	ACK – OK (Issued to LFAST from SIPI)	32	Tx
09	ACK – Fault (Issued to LFAST from SIPI)[1]	32	Tx
0A	Read Answer – OK (Issued to LFAST from SIPI)	64	Tx
0B	Reserved	—	—
0C	Unimplemented – Trigger comm and with ACK	32	—
0D	Reserved	—	—
…	…	…	…
11	Reserved	—	—
12	ID Register Read Request	32	Rx
13	Reserved	—	—
…	…	…	…
16	Reserved	—	—
17	Unimplemented – Stream Data with ACK (32 bytes)	288	—
18	Reserved	—	—
…	…	…	…
1F	Reserved	—	—

1. ACK – Fault is also referred to as “NACK” and “ACK Error” in the HSSL/SIPI Interprocessor Bus Protocol Specification.

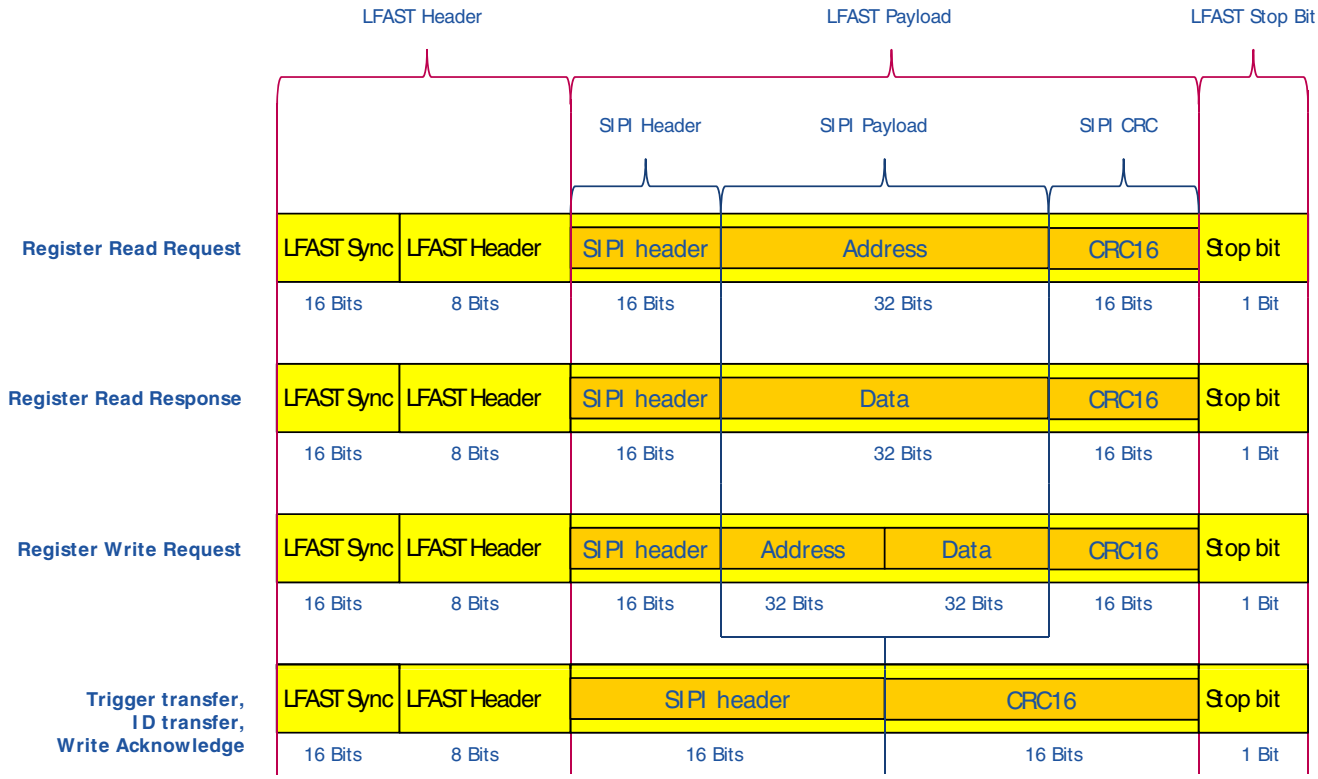


Figure 74-6. Zipwire message format

NOTE

Identification Register Read is the same as the Register Read response format.

74.8.1 ED power supply constraints

For ED devices the following power domain user constraints must be observed:

- The Vdd_hv_jtag_osc and Vdd_hv_io_bd supplies need to be connected on the application PCB.
- For cold start operation where a tool must be able to communicate via JTAG and read/write ED RAM contents, Vdd_lv_stby, Vdd_hv_jtag_osc and Vdd_hv_io_bd must remain powered.
- For cold start operation where only ED RAM contents have to be maintained, only Vdd_lv_stby must remain powered.

74.9 Debug Zipwire interconnections

Debug LFAST has the following connections:

- Tx and Rx connections to the pads via the SIUL and the MSCR registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.
- Tx Data Port/Rx Data Port — Connected to the LFAST Switch. This allows LFAST data to be dynamically routed to/from either Debug SIPI for Debug Zipwire operation or JTAGM for JTAG over LFAST operation.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

LFAST Switch has the following connections:

- Tx Data Port/Rx Data port directly connected to Debug SIPI.
- Tx Data Port/Rx Data port directly connected to JTAGM.
- Tx Data Port/Rx Data port directly connected to Debug LFAST.
- LFAST switch has no IPS interface; no registers; no interrupts.

Debug SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers. For the Debug SIPI, a restricted set of registers is implemented. In use, no configuration of these registers is needed from either s/w or a connected tool, as the default settings should be usable for typical operation.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.

74.10 Debug Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Inititator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

Bandwidth is the rate at which data can be read or written between the tool and the MCU. It assumes that read or write commands from the Inititator node (the Tool), can be sent continuously at the highest speed the Target node can consume those commands.

Latency is the time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

74.10.1 Read performance

A Zipwire read operation consists of three stages:

- Inititator sends SIPI Read command to Target.
- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data

- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in a LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes 278 ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of 6 × slow XBAR cycles to read different memories.
- Slow XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, time for SIPI to read an address location is:

- (13 × 50 MHz clocks) + (6 × 100 MHz clocks) = 320 nS

74.10.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

Read Bandwidth = 4 bytes in 320 ns = 12.5 MB/second

74.10.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

Read Latency = 278 ns + 320 ns + 278 ns = 876 ns

74.10.2 Write performance

A Zipwire write operation consists of three stages:

- Inititator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address
- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header
- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32-bit (acknowledge)
- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes 378 ns
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes 178 ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of 5 × slow XBAR cycles to write different memories.
- Slow XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, the time for SIPI to write an address location is:

- (13 × 50 MHz clocks) + (5 × 100 MHz clocks) = 320 ns

74.10.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = 4 bytes in 378 ns = 10.6 MB/second

74.10.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = 378 ns + 310 ns + 178 ns = 866 ns

Chapter 75
Debug Serial Interprocessor Interface (SIPI)

75.1 Introduction

The Debug Serial Interprocessor Interface (SIPI) is an application layer protocol which runs on top of the Debug LFAST (LVDS Fast Asynchronous Serial Transmission) module. It is used by the local device to access the shared memory of a remote device. Debug SIPI defines point-to-point full duplex communication between two nodes, where LFAST works as a physical medium of communication between both the devices.

Debug SIPI implements the Target mode of a communication protocol in hardware. Debug SIPI can interpret 'read' and 'write' commands received from an LFAST module and convert them into bus accesses to local memory; then send back to the LFAST module either read data or a write acknowledge response. This provides a mechanism for an external device, such as an external tool, to read and write the address map of the local MCU. The bandwidth of these read/write commands is approximately 10 MBytes/second when accessing random addresses using 32-bit transfers.

75.2 Overview

Debug SIPI can access memory mapped resources directly through its AHB master interface. SIPI have four channels, 0, 1, 2, 3. Payload width for all channels is 32 bits.

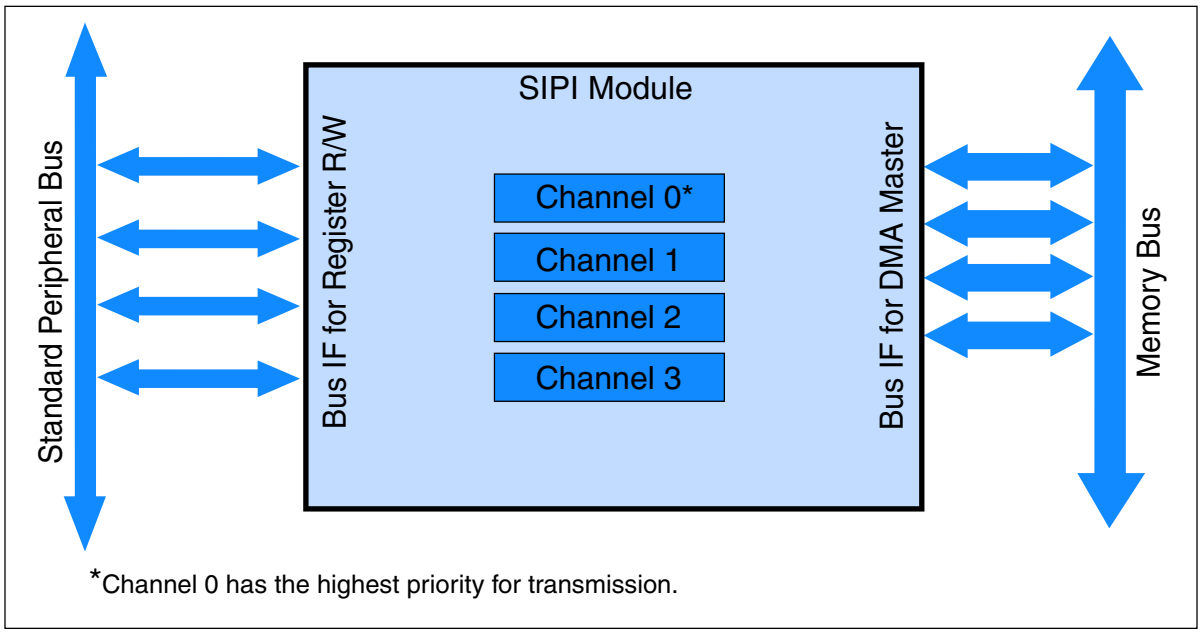
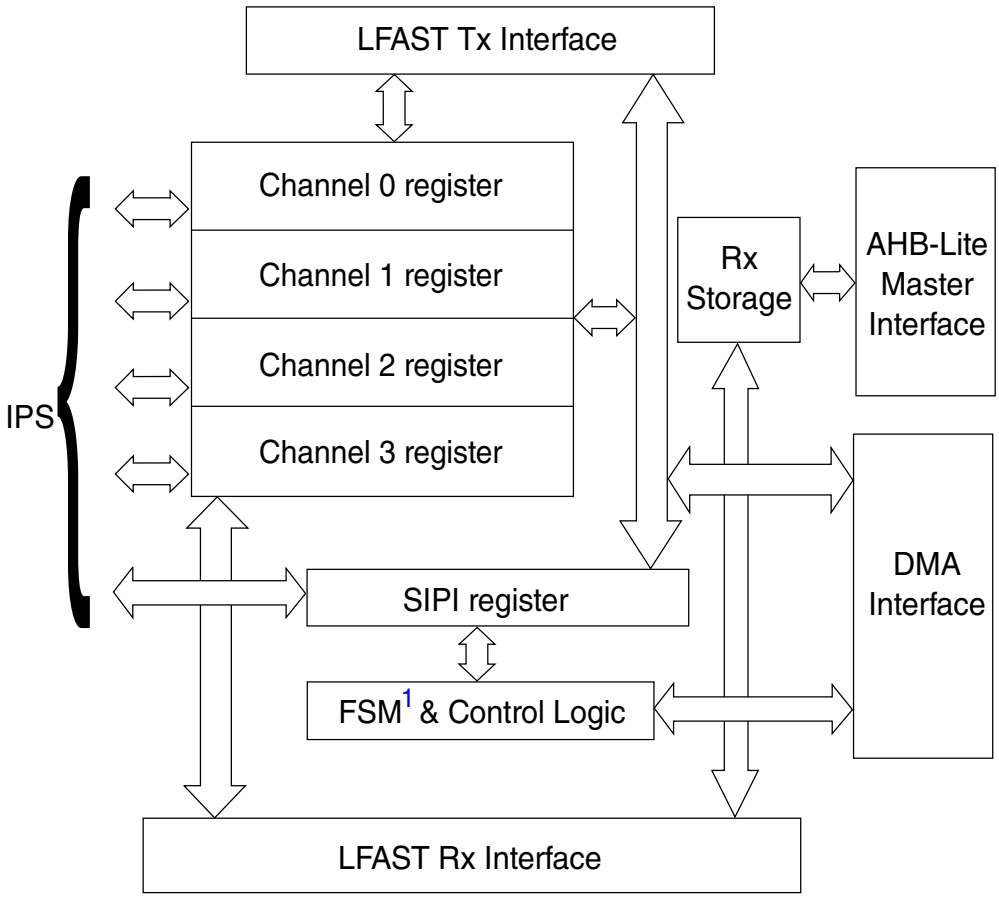


Figure 75-1. SIPI module

75.3 Debug SIPI block diagram



[1] Finite State Machine

Figure 75-2. SIPI block diagram

75.4 Feature description

This section describes the features of the Debug SIPI module.

75.4.1 Main features

- Supports 8, 16 and 32-bit read and write commands.
- Command/response includes a 16-bit CRC to ensure data integrity.
- Command/response includes a 2-bit tag to allow tracking of data to commands in a pipelined system.

- Command/response includes a 3-bit channel identifier to allow multiple Initiators to send commands to a single target node.
- Command/response includes a 16-bit header that defines, the function, channel and tag.
- Every command generates a response to allow a 'sliding window' flow control protocol to be implemented.
- AHB master interface that is used by target node to access memory mapped resources.

75.4.2 Standard features

- IPS bus interface (PBRIDGE)
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

75.5 Debug SIPI operation from reset

When the Debug SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

75.6 Functional description

75.6.1 External signals

The Debug SIPI has no chip external signals.

75.6.2 Frame format

All frames have the same general format:

- 16 bit header

- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID. The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

75.6.2.1 Register read request

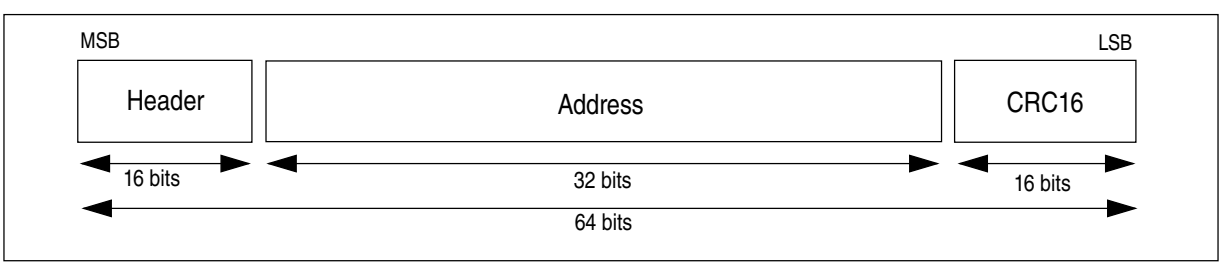


Figure 75-3. SIPI register read request

75.6.2.2 Register read response, ID request response

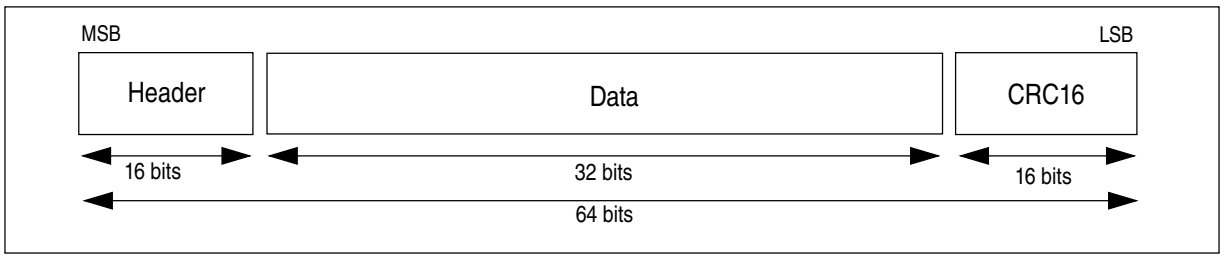


Figure 75-4. SIPI read response

75.6.2.3 Register write request

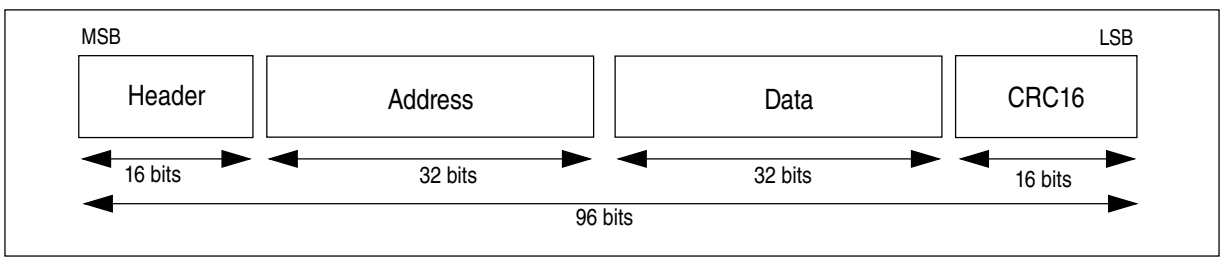


Figure 75-5. SIPI register write request

75.6.2.4 ID transfer, write acknowledge

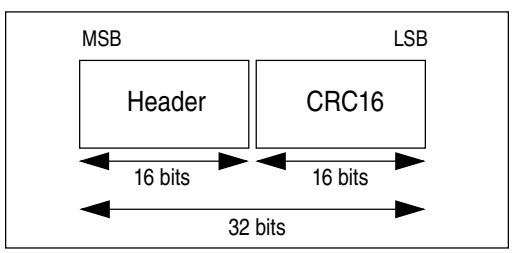


Figure 75-6. SIPI write acknowledge

75.6.2.5 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

75.6.2.5.1 Debug SIPI header coding

Figure 75-7, Table 75-1, and Figure 75-8 show how the Debug SIPI header bits are coded.

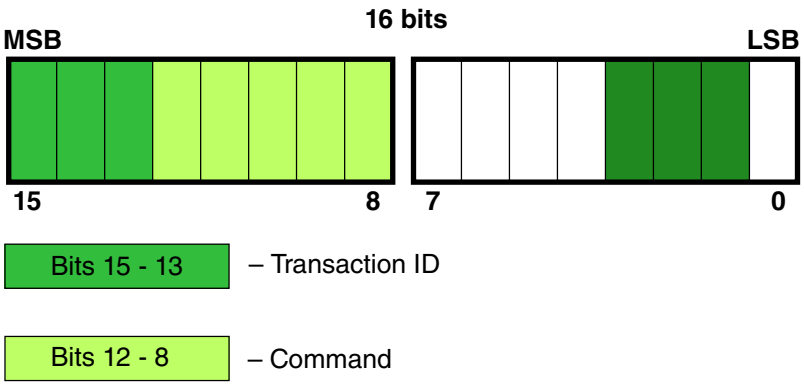


Figure 75-7. Debug SIPI header coding

Table 75-1 shows the command coding for the bits[12:8] of the header.

Table 75-1. Debug SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Reserved	—
0D	Reserved	—
…	…	…
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
…	…	…
16	Reserved	—
17	Reserved	—
18	Reserved	—
…	…	…
1F	Reserved	—

Figure 75-8 shows the channel number field in the header.

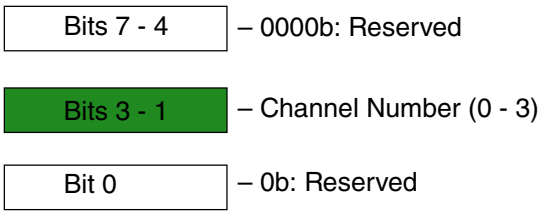


Figure 75-8. Debug SIPI header channel field

75.6.2.6 Address field

The address field is 32 bits wide with the MSB transmitted first.

75.6.2.7 Payload field

Table 75-2 shows the payload sizes of LFAST frames.

Table 75-2. Payload size of LFAST channel frame

LFAST Code	Debug SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 75-3 shows the converted coding of LFAST for a given Debug SIPI code.

Table 75-3. Converted coding of LFAST channel code for Debug SIPI headers

LFAST Code	Debug SIPI Code	Channel (hex)[1]
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by Debug SIPI)
1001b	001b	F (not used by Debug SIPI)
1010b	010b	G (not used by Debug SIPI)
1011b	011b	H (not used by Debug SIPI)

1. Debug SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
   Debug SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
   Debug SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
   Debug SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

75.6.2.8 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). MSB is sent first in the data stream.

75.6.2.8.1 CRC field examples

75.6.2.8.1.1 Example 1 – 32 bit write

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

75.6.2.8.1.2 Example 2 – 32 bit read

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

75.6.2.8.1.3 Example 3 – Event command

- Header = 6C0Eh
- CRC = B286h

75.7 Transfer types

This section describes the available transfer types of the Debug SIPI module. The Debug SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

75.7.1 Read transfer

A Read transfer can be of two types:

- Read Request Transfer (at initiator node)
- Read Response Transfer (at target node)

75.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in Figure 75-9.

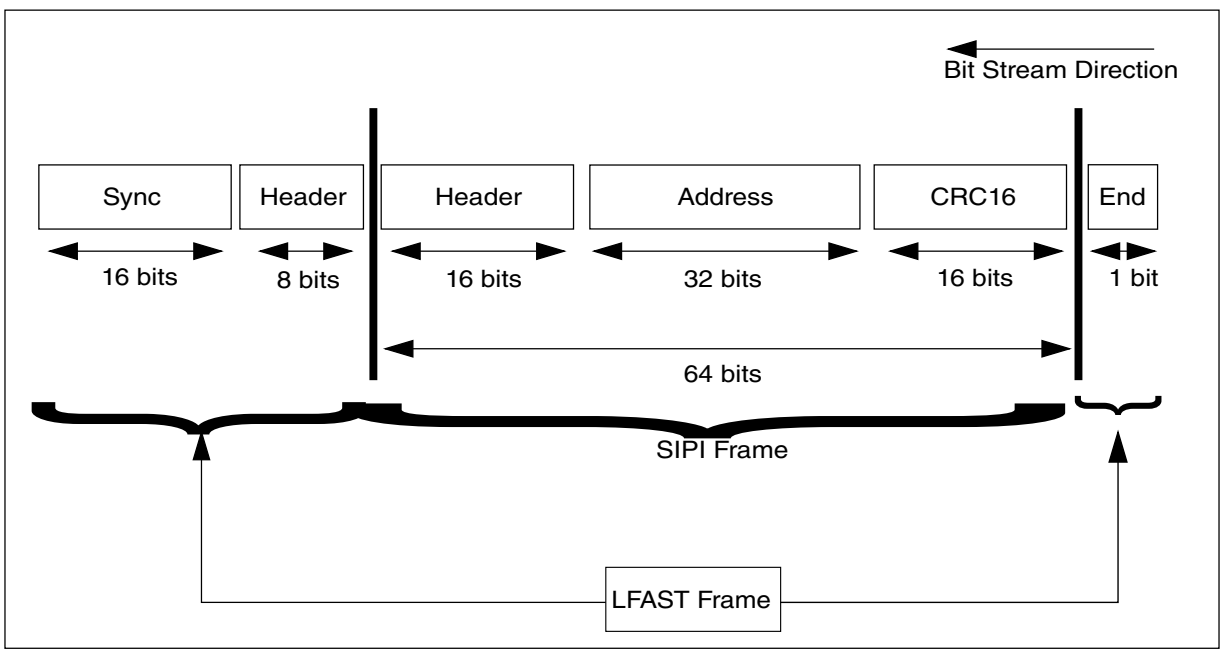


Figure 75-9. Read request transfer

75.7.2 Register read answer transfer

In response to a read request, Debug SIPI sends header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see Figure 75-10). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see Figure 75-11).

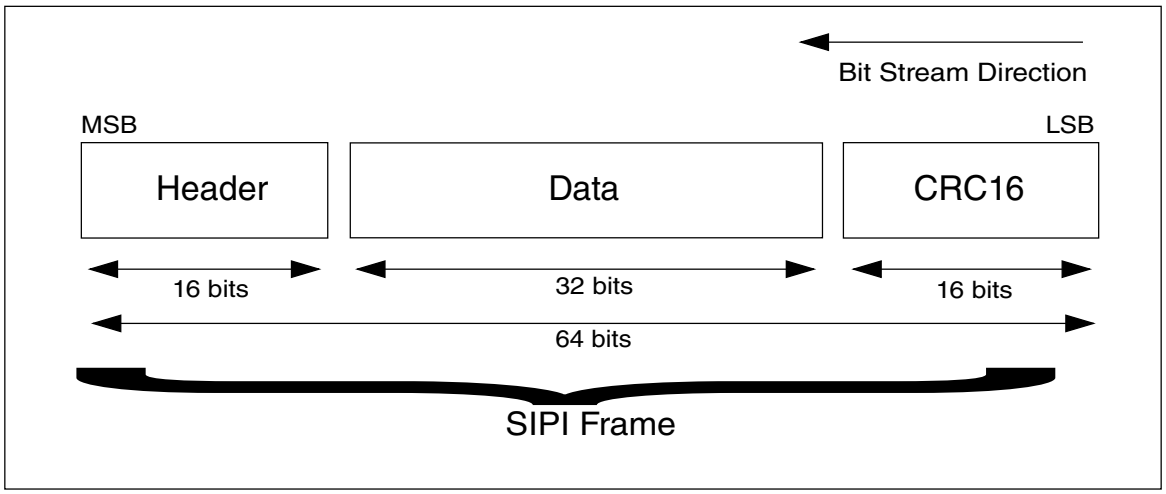


Figure 75-10. Read answer transfer

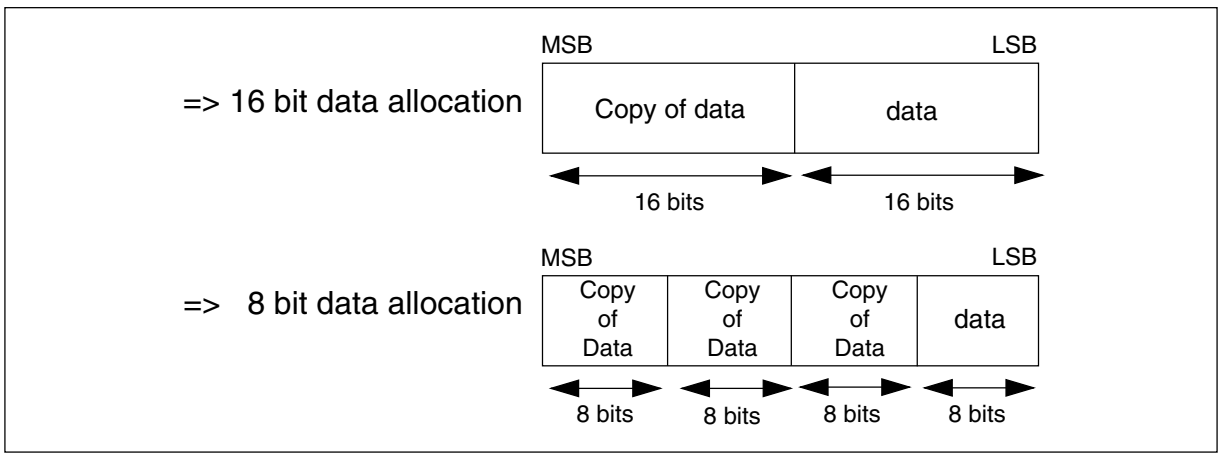


Figure 75-11. Data allocation

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)

- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as halfword enable.
  - Bit[0]:
    - 0 - halfword 1 enabled (MSB)
    - 1 - halfword 0 enabled (LSB)

75.7.3 Register Write transfer

Register Write transfer can be of type:

- Normal write transfer - channels 0, 1, 2 and 3

A Register Write transfer can be done through Normal Write transfer on channels 0, 1, 2 and 3 (see Normal write transfer) as shown in Figure 75-12.

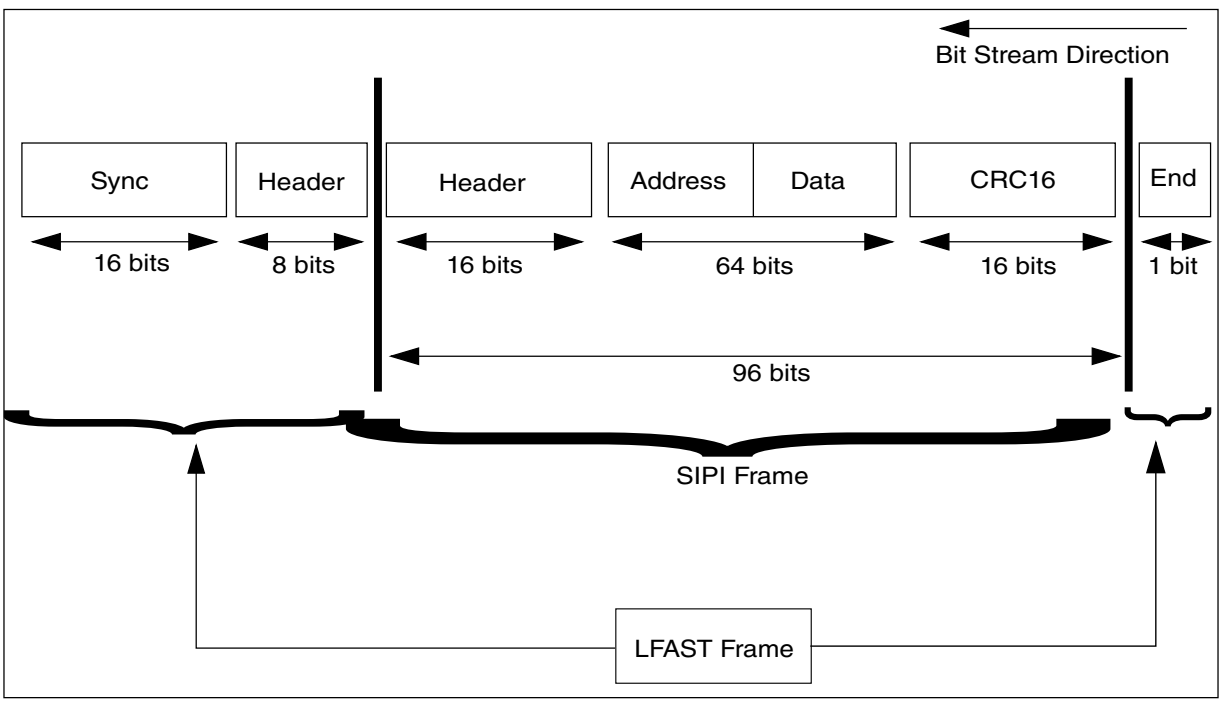


Figure 75-12. Register write transfer (showing LFAST frame encapsulation)

75.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and CRC as shown in Figure 75-13.

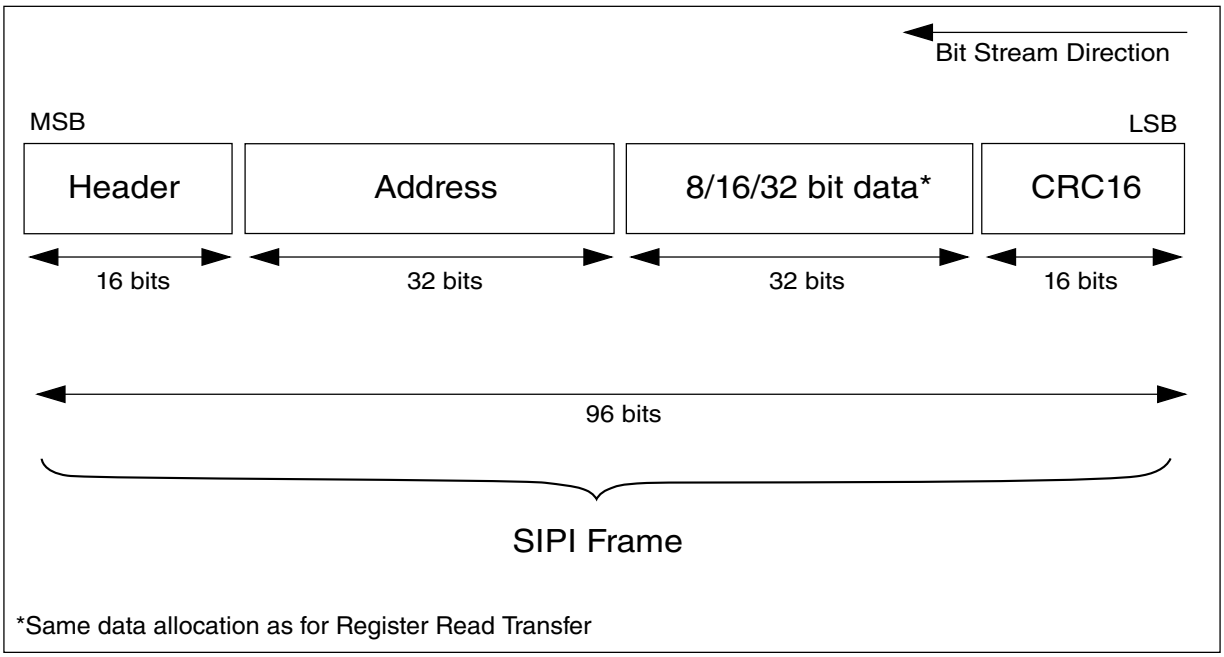


Figure 75-13. Write transfer (LFAST frame encapsulation is not shown)

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
  - Bit[0]:
    - 0 - half-word 1 enabled (MSB)
    - 1 - half-word 0 enabled (LSB)

75.7.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see Write Aknowledge transfer. The CRC bits are calculated on the header field.

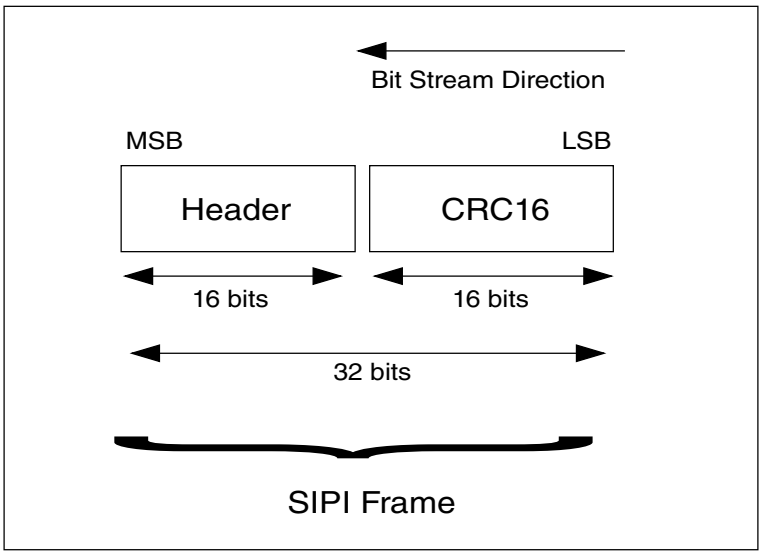


Figure 75-14. Write Aknowledge transfer (LFAST encapsulation is not shown)

75.7.5 ID request response

75.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in Figure 75-15.

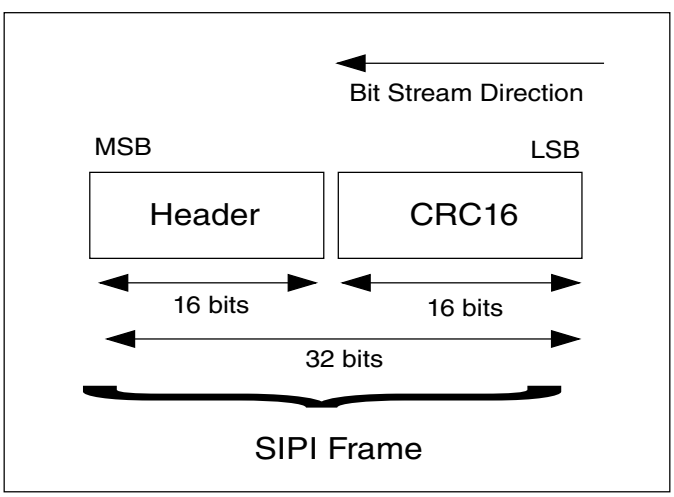


Figure 75-15. ID request transfer (LFAST encapsulation is not shown)

75.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in Figure 75-16.

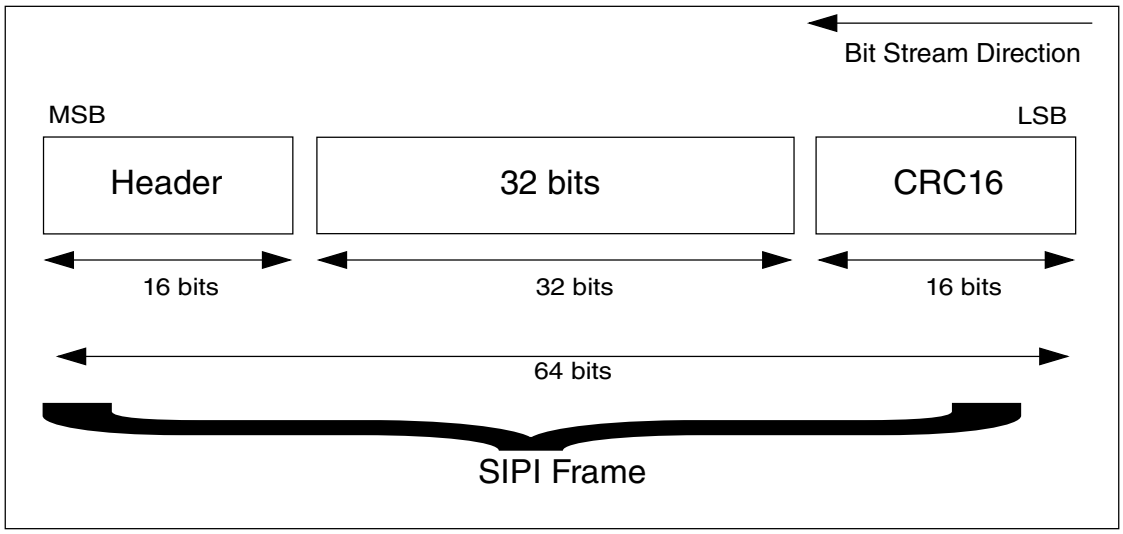


Figure 75-16. ID request response transfer

Note

The ID request response contains the value of the CHIP ID in place of data.

75.8 Transfer API and flow charts

On a single Write transfer request reception (Figure 75-17):

1. Debug SIPI will place the address, data and control information on its AHB master interface.
2. When the process completes, the Debug SIPI will generate an acknowledge frame and send it back to the LFAST.

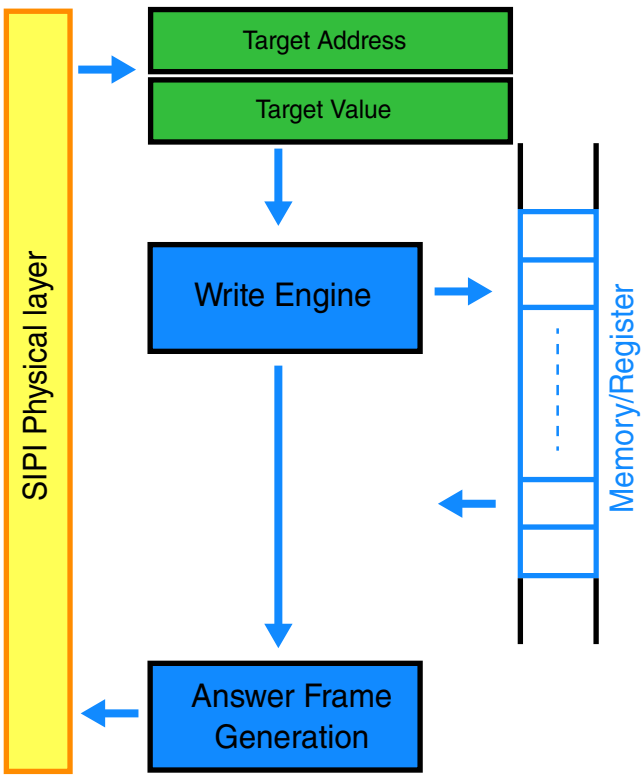


Figure 75-17. SIPI single register write API – Flow Chart

On Multiple Write transfer request reception:

1. Debug SIPI will place the address, data and control information on its AHB Master Interface.
2. When the process completes, Debug SIPI will generate an acknowledge frame and send it back to the LFAST.

On Single Read transfer request reception:

1. Debug SIPI will place the address and control information on its AHB Master Interface.
2. When the process completes, Debug SIPI will send read response back to LFAST.

On Multiple Read transfer request reception:

1. Debug SIPI then will start reading data through its AHB interface.
2. When the transfer is completed and all data registers are full, it will calculate CRC and send the response frame back to LFAST.

75.9 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

1) The CRC seed is initialized by FFFF_FFFFh.

2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h.

4) Address will be sent as 22CC_4488h.

5) Data will be sent as FF77_BB33h.

75.10 Memory map and register definition

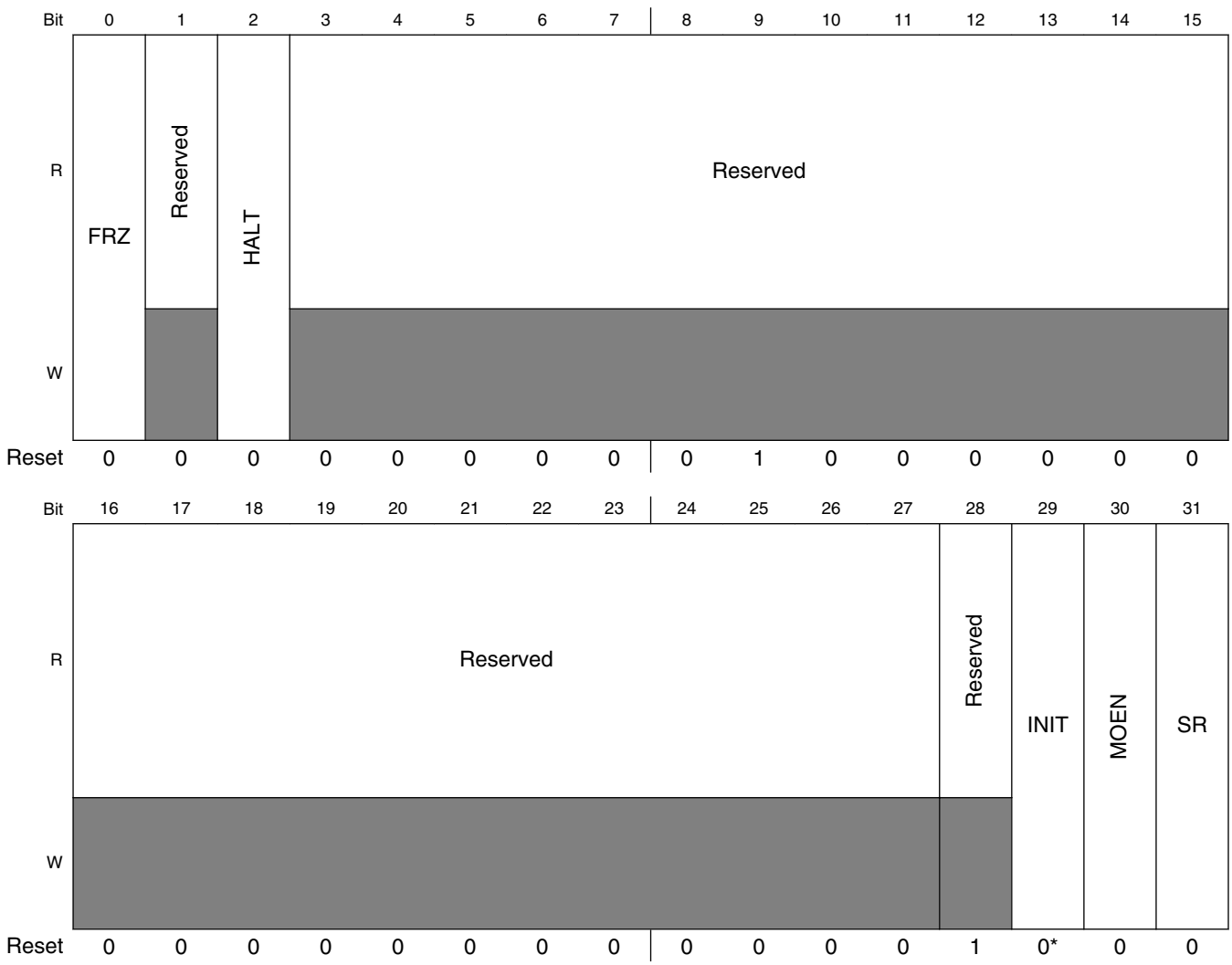
SIPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C	SIPI Module Configuration Register (SIPI_MCR)	32	R/W	See section	75.10.1/ 3998
A0	SIPI Status Register (SIPI_SR)	32	R	0000_0000h	75.10.2/ 4000

75.10.1 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a global 32-bit configuration register.

Address: 0h base + 9Ch offset = 9Ch



* Notes:
- INIT field: Can only be written when SIPI_MCR[MOEN] = 1.

SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode.

Table continues on the next page...

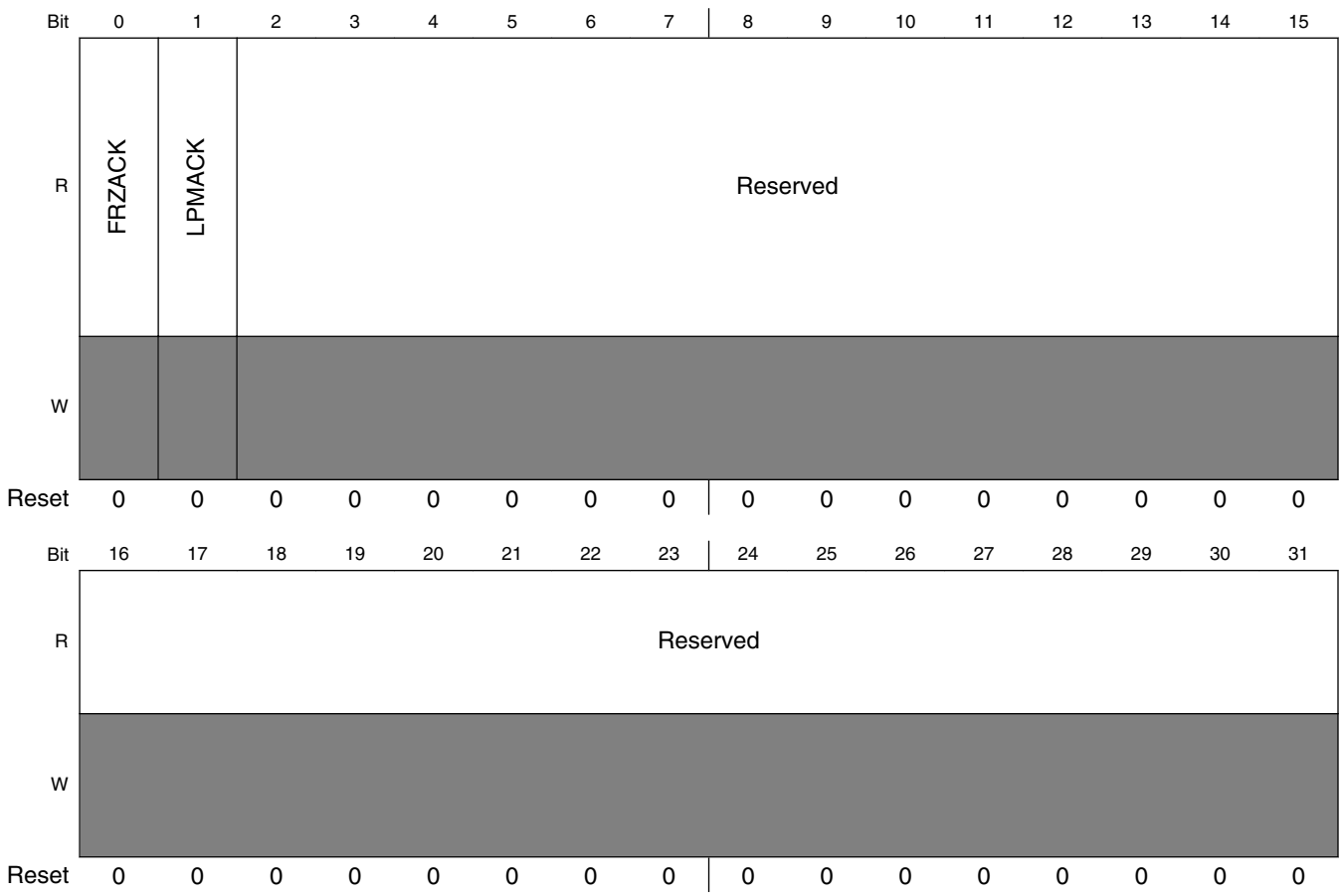
SIPI_MCR field descriptions (continued)

Field	Description
	NOTE: Can only be written when SIPI_MCR[MOEN] = 1. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1 Reserved	This field is reserved.
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–27 Reserved	This field is reserved.
28 Reserved	Target Enable This field is reserved.
29 INIT	Initialization Mode Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits can't be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, all SIPI operations are immediately stopped.
31 SR	Soft Reset Setting this bit clears all status and error registers, and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

75.10.2 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Address: 0h base + A0h offset = A0h



SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited. 0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge.

Table continues on the next page...

SIPI_SR field descriptions (continued)

Field	Description
	This read-only bit indicates that SIPI is in Disable Mode. Either of these low power modes can not be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI is not in low power mode 1 SIPI is either in Disable Mode
2–31 Reserved	This field is reserved.

Chapter 76
LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug

76.1 Introduction

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in slave only mode enabling high speed debug communications through the JTAG port.

76.2 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

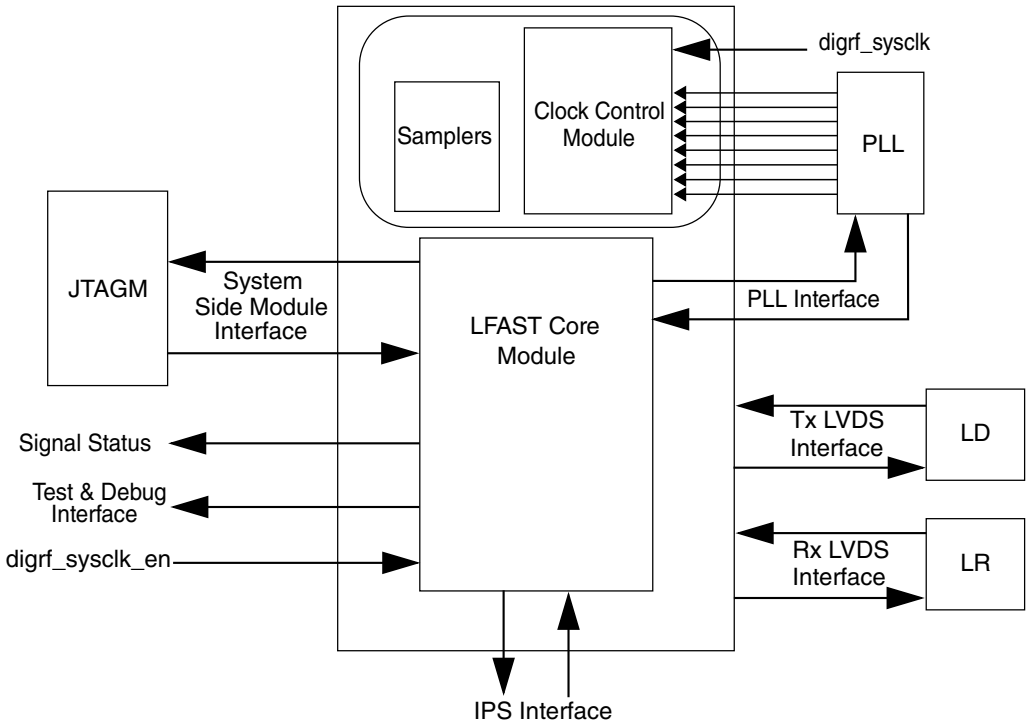


Figure 76-1. LFAST block diagram

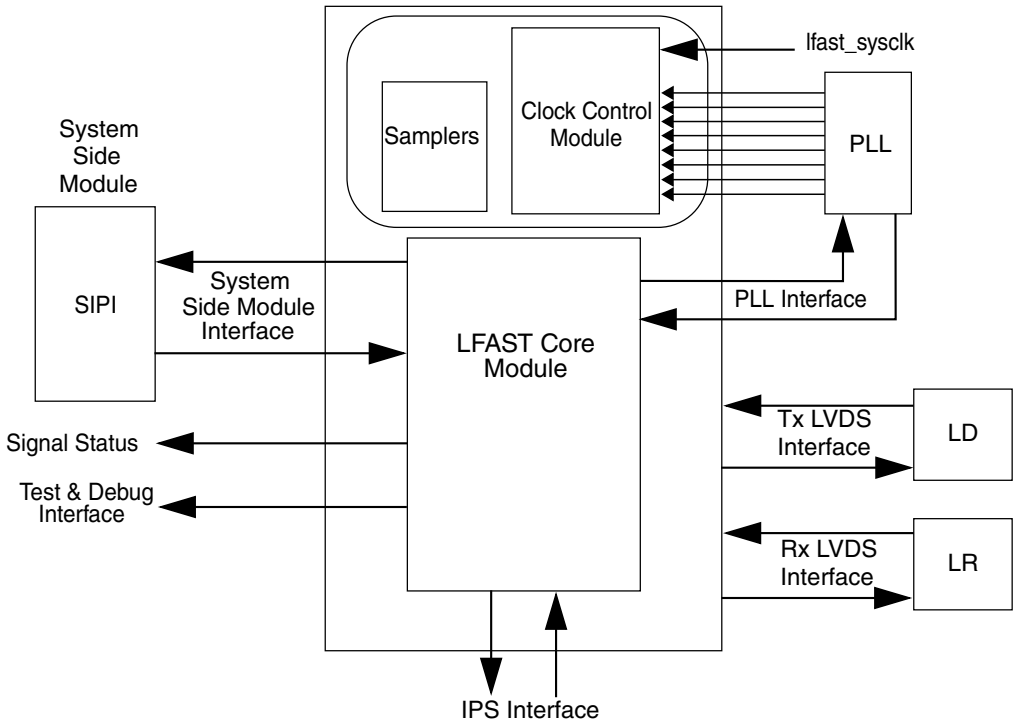


Figure 76-2. LFAST block diagram

76.3 External signals

LFAST is a six pin interface with the following signals after enabling high speed debug mode:

- lfast_sysclk_en
  - Interface enable signal in LFAST Slave Only mode
- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface with maximum data rate of 320 Mbps.

76.3.1 LFAST operating data rates

Table 76-1. Operating modes for LFAST interface

Interface	Tx/Rx Data Rate
LFAST Transmit	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps
LFAST Receive	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low and data rates 312 Mbps and 320 Mbps is referred to as high data rates.

76.4 LFAST frame structure

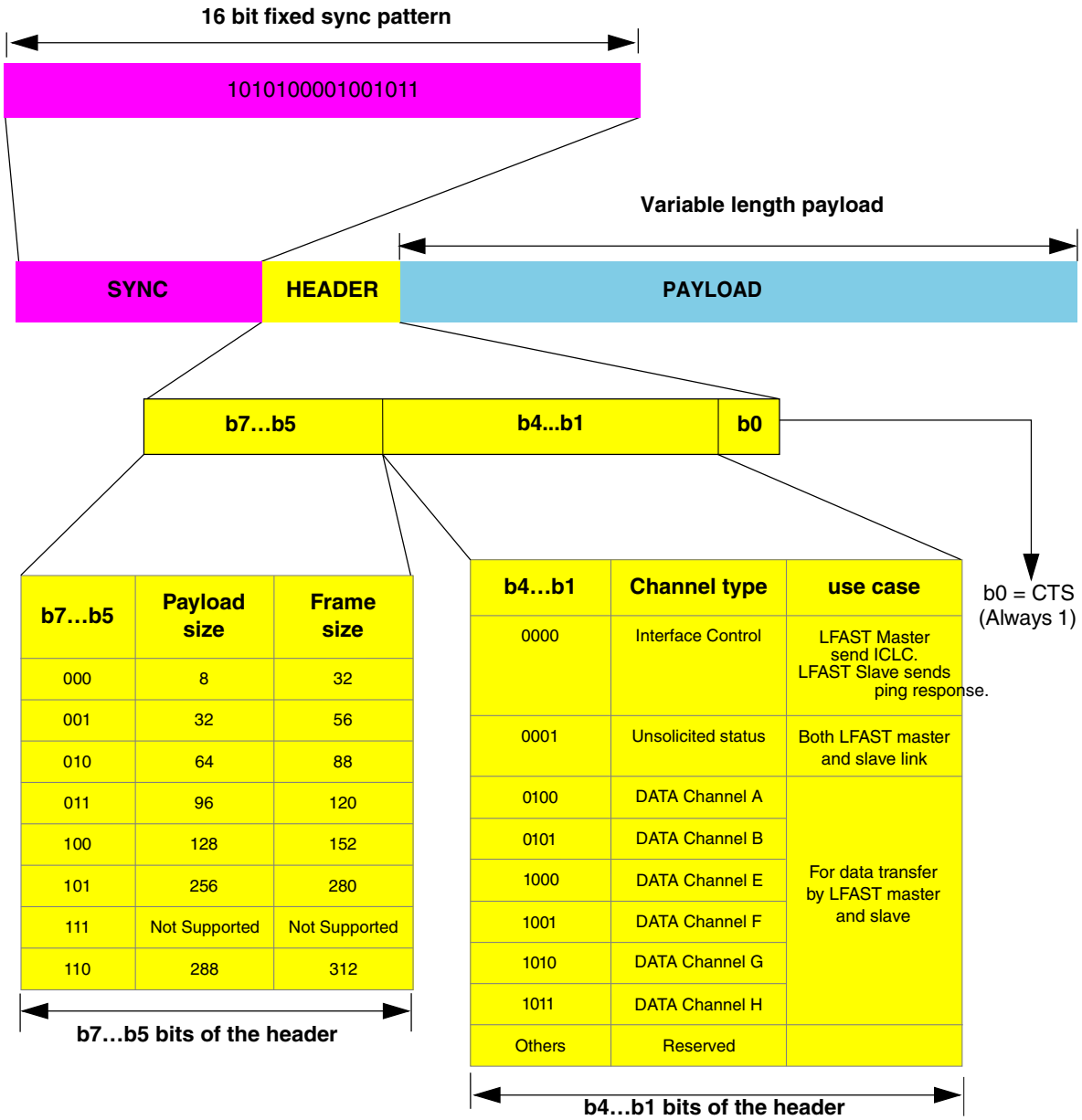
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See Figure 76-3 for details of the LFAST frame.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	Not Supported	Not Supported
110	288	312

b4…b1	Channel type	use case
0000	Interface Control	LFAST Master send ICLC. LFAST Slave sends ping response.
0001	Unsolicited status	Both LFAST master and slave link
0100	DATA Channel A	For data transfer by LFAST master and slave
0101	DATA Channel B	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 76-3. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

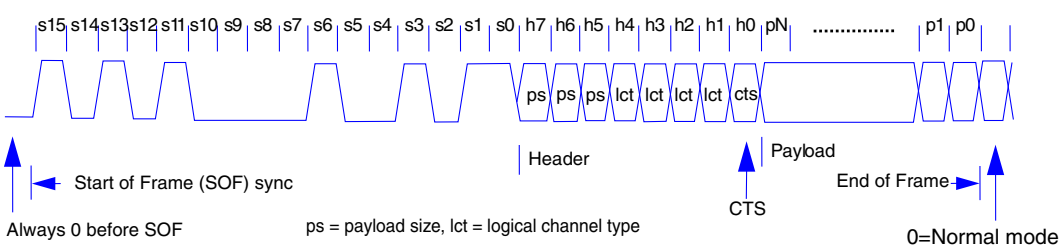


Figure 76-4. Serial frame structure

The frame structure is shown in Figure 76-4 and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in Table 76-2 :

Table 76-2. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	288	312
111	—	—

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type.
- **Header - (b0):** CTS on the both LFAST master and slave devices. In slave only mode the CTS field will always be 1.
- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

76.5 Features

- Supports slave only mode.
- Supports asynchronous data transfer at data rates up to a maximum of 320 Mbps.

- Transmits and receives data, ICLC and unsolicited frames.
- Receives ICLC frames
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation .
- Supports for detection of unsupported channel number and unsupported payload size.

76.6 Memory map and register definition

All registers are 32 bits wide.

LFAST memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	76.6.1/4011
4	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	76.6.2/4013
8	LFAST Correlator Control Register (LFAST_COCR)	32	R/W	0000_000Eh	76.6.3/4014
C	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	76.6.4/4016
10	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	76.6.5/4017
14	LFAST Rate Change Delay Control Register (LFAST_RCDCR)	32	R/W	000F_0000h	76.6.6/4018
18	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	76.6.7/4018
20	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	76.6.8/4020
40	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	76.6.9/4021
44	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	76.6.10/ 4023

Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
48	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	76.6.11/ 4024
4C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	76.6.11/ 4024
50	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	76.6.11/ 4024
54	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	76.6.11/ 4024
58	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	76.6.11/ 4024
5C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	76.6.11/ 4024
60	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	76.6.11/ 4024
64	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	76.6.11/ 4024
68	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	76.6.11/ 4024
80	LFAST Global Status Register (LFAST_GSR)	32	R	See section	76.6.12/ 4025
94	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	76.6.13/ 4026
98	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	76.6.14/ 4027
9C	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	76.6.15/ 4029
A0	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	R/W	0000_0000h	76.6.16/ 4031
A4	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	76.6.17/ 4033
A8	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	76.6.18/ 4034
AC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	76.6.19/ 4035
B0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	76.6.19/ 4035
B4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	76.6.19/ 4035
B8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	76.6.19/ 4035
BC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	76.6.19/ 4035
C0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	76.6.19/ 4035

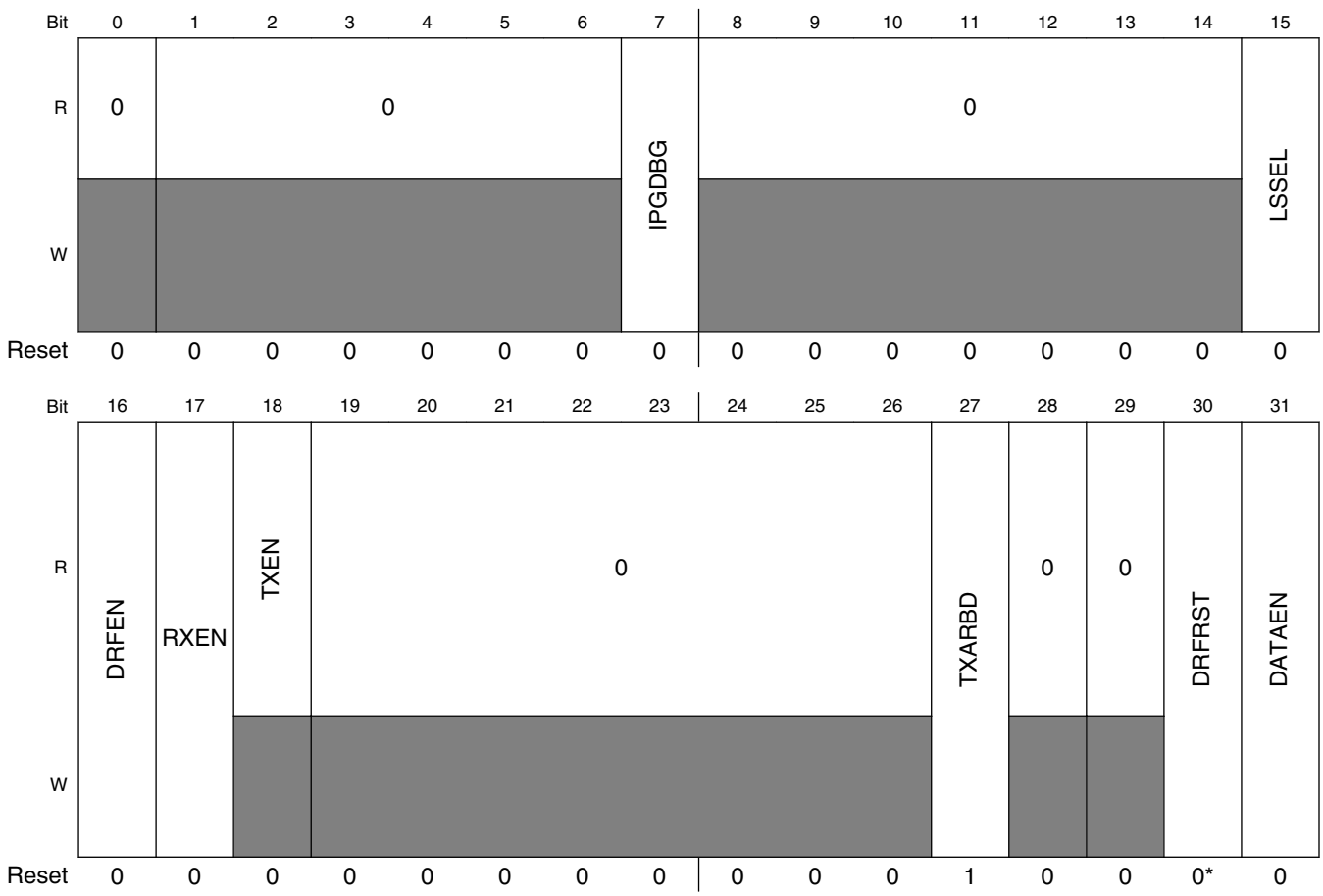
Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	76.6.19/ 4035
C8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	76.6.19/ 4035
CC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	76.6.19/ 4035

76.6.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 1h base + 0h offset = 1h



* Notes:

- DRFRST field: Set by user software and then cleared by system hardware.

LFAST_MCR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit shows the whether the transmitter is enabled or disabled. This field is read only in slave only mode (LFAST_GSR[DUALMD] = 0) and must not be written. 0 LFAST transmitter Interface is disabled. 1 LFAST transmitter Interface is enabled.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset..

Table continues on the next page...

LFAST_MCR field descriptions (continued)

Field	Description
	0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

76.6.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 1h base + 4h offset = 5h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															DRMD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							RDR	0							TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

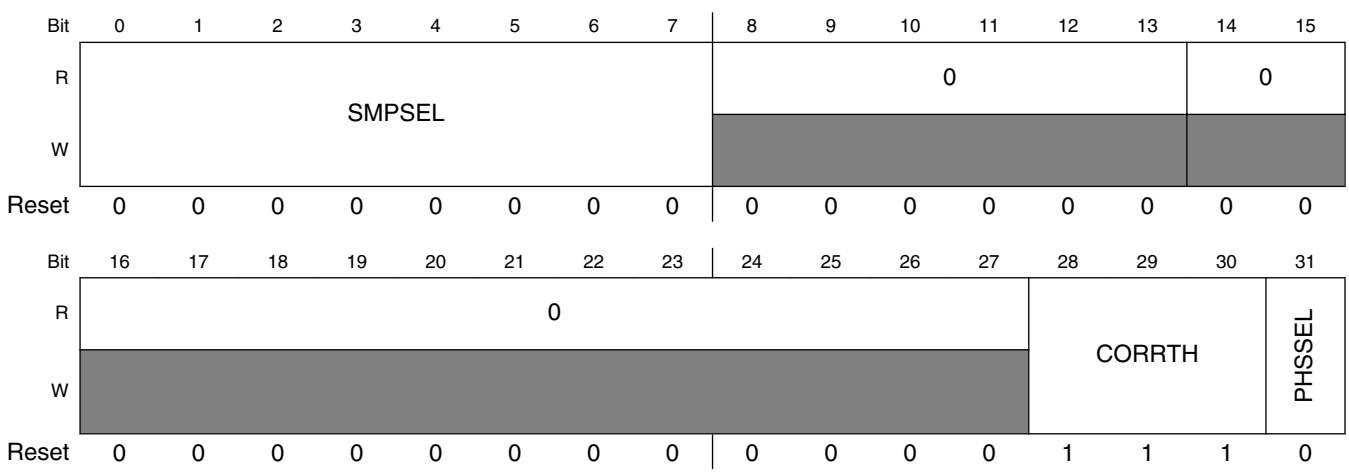
LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. This bit shouldn't be modified by S/W and left as SCR[DRMD] =1.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit shows the data rate of the receiver. This bit should not be modified by S/W in slave only mode (LFAST_GSR[DUALMD] = 0). 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is 312/320 Mbps.
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit shows the data rate of the transmitter. This bit should not be modified by S/W in slave only mode (LFAST_GSR[DUALMD] = 0). 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is 312 /320 Mbps.

76.6.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 1h base + 8h offset = 9h



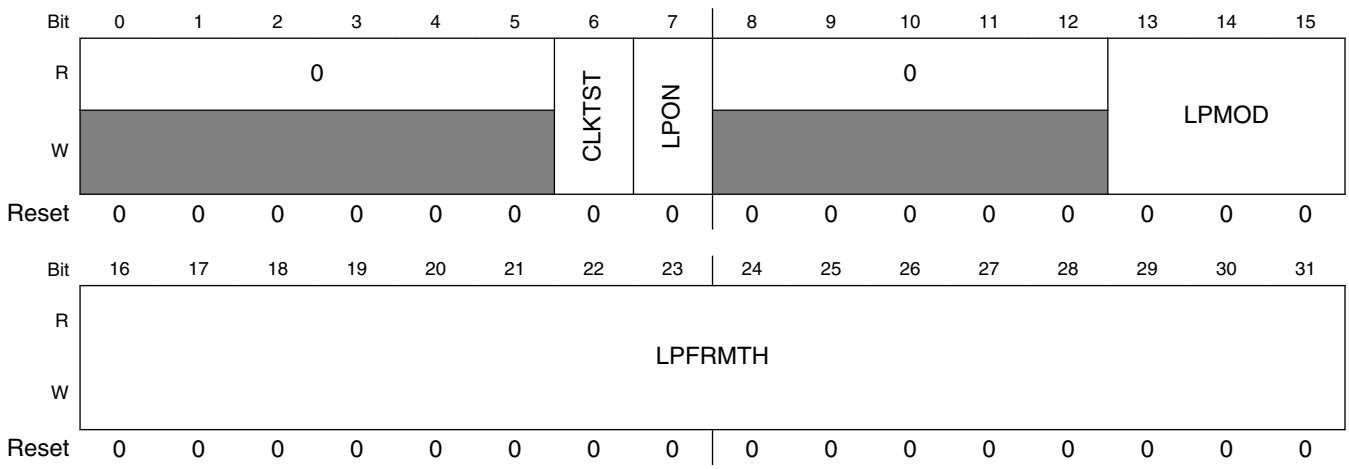
LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0. 00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

76.6.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 1h base + Ch offset = Dh



LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by H/W on reception of an ICLC command. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved

Table continues on the next page...

LFAST_TMCR field descriptions (continued)

Field	Description
	110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

76.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 1h base + 10h offset = 11h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															LPCNTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload

Table continues on the next page...

LFAST_ALCR field descriptions (continued)

Field	Description
	... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

76.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 1h base + 14h offset = 15h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DRCNT				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

76.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 1h base + 18h offset = 19h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT			
W																																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ...

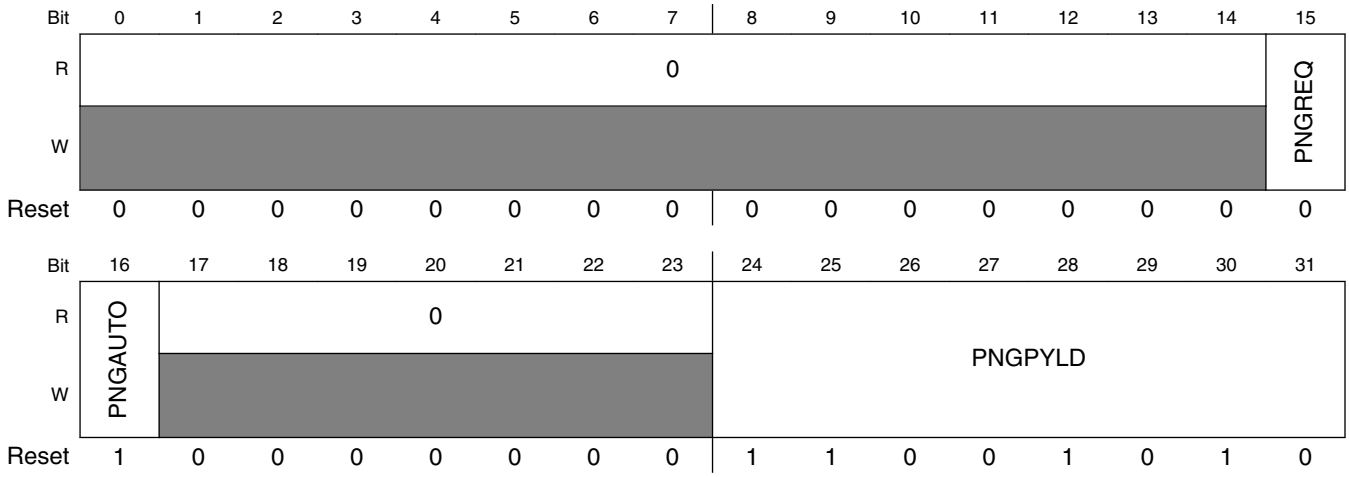
Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
	12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 µs) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 µs) ... Eh 14 cycles Fh 15 cycles

76.6.8 LFAST Ping Control Register (LFAST_PICR)

Address: 1h base + 20h offset = 21h

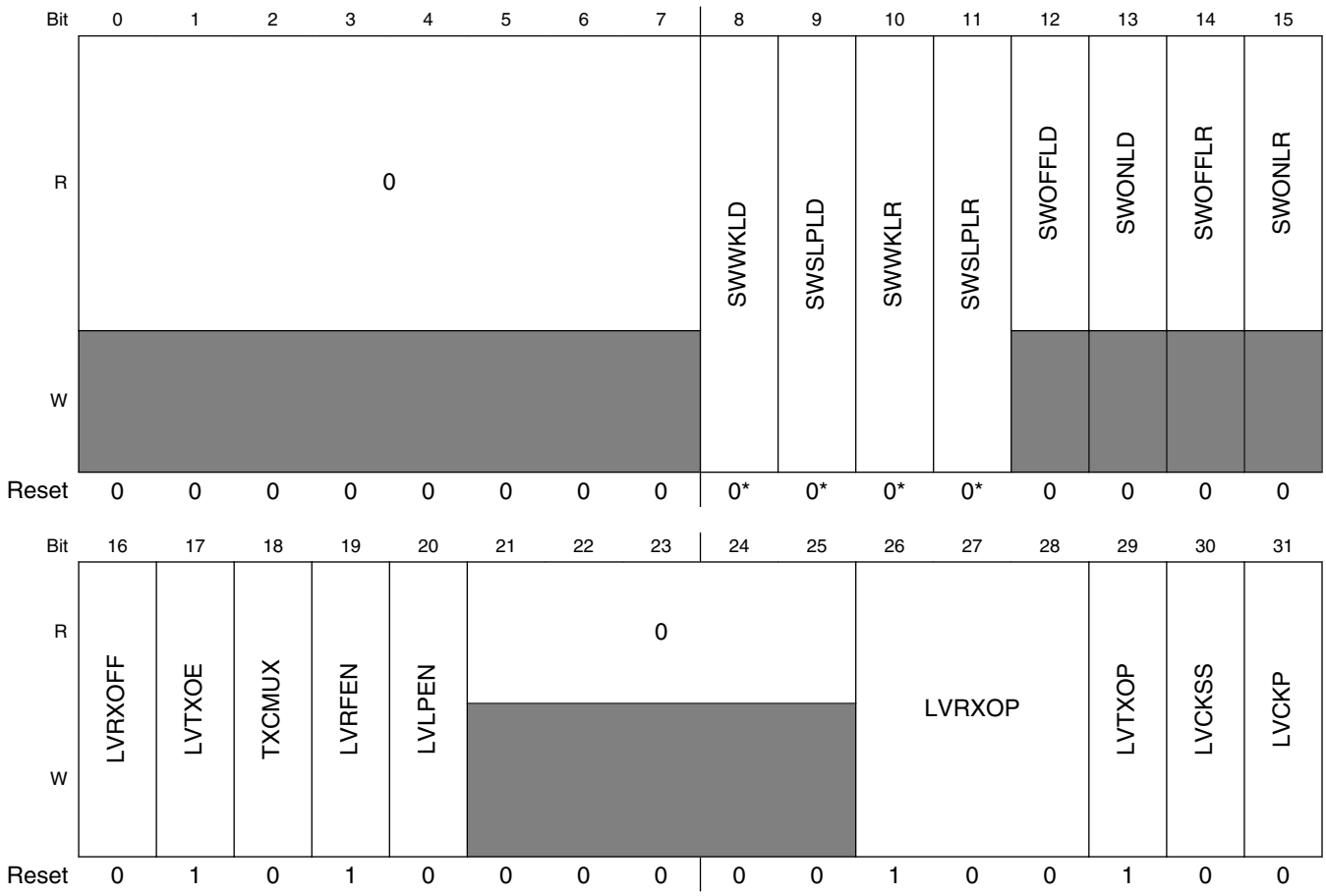


LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. This bit should not be modified by S/W.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0.

76.6.9 LFAST LVDS Control Register (LFAST_LCR)

Address: 1h base + 40h offset = 41h



* Notes:
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field.

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
	0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
13 SWONLD	SW signal to turn ON the LVDS LD. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
15 SWONLR	SW signal to turn ON the LVDS LR. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled

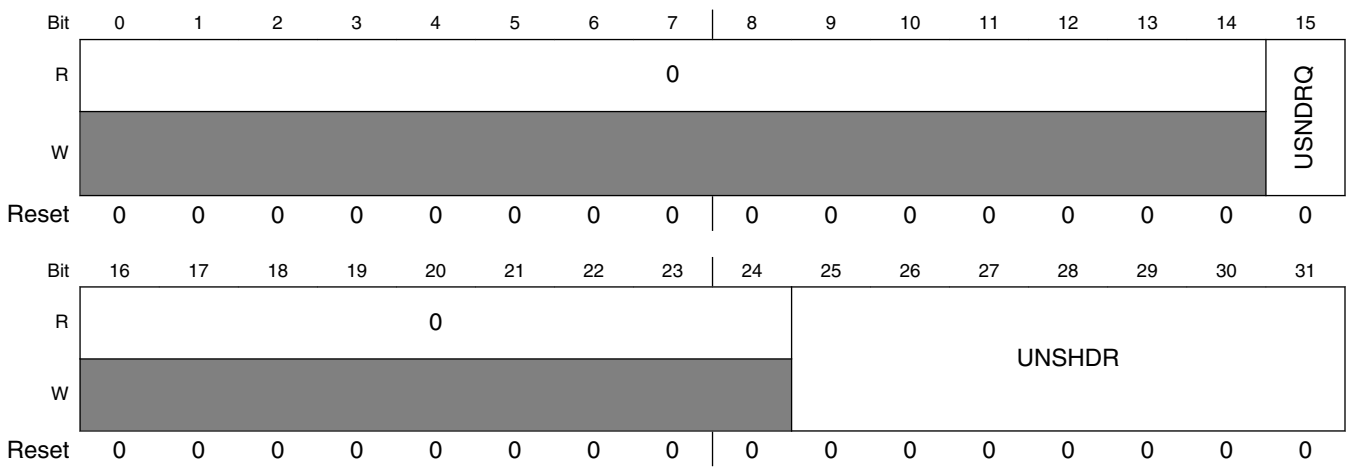
Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–28 LVRXOP	LVRXOP[2] disables termination resistance: 0 Disable termination 1 Enable termination LVRXOP[1] Not used LVRXOP[0] Defines the LVDS LR Bias current reference
29 LVTXOP	Control signal for LFASTand Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Data select. This bit is used for selecting use of clock sampled data or normal data in LVDS. 0 normal data to be used inside the LVDS 1 synchronized data to be used inside the LVDS
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

76.6.10 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 1h base + 44h offset = 45h



LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_UNSTCR field descriptions (continued)

Field	Description
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

76.6.11 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)

Address: 1h base + 48h offset + (4d × i), where i=0d to 8d

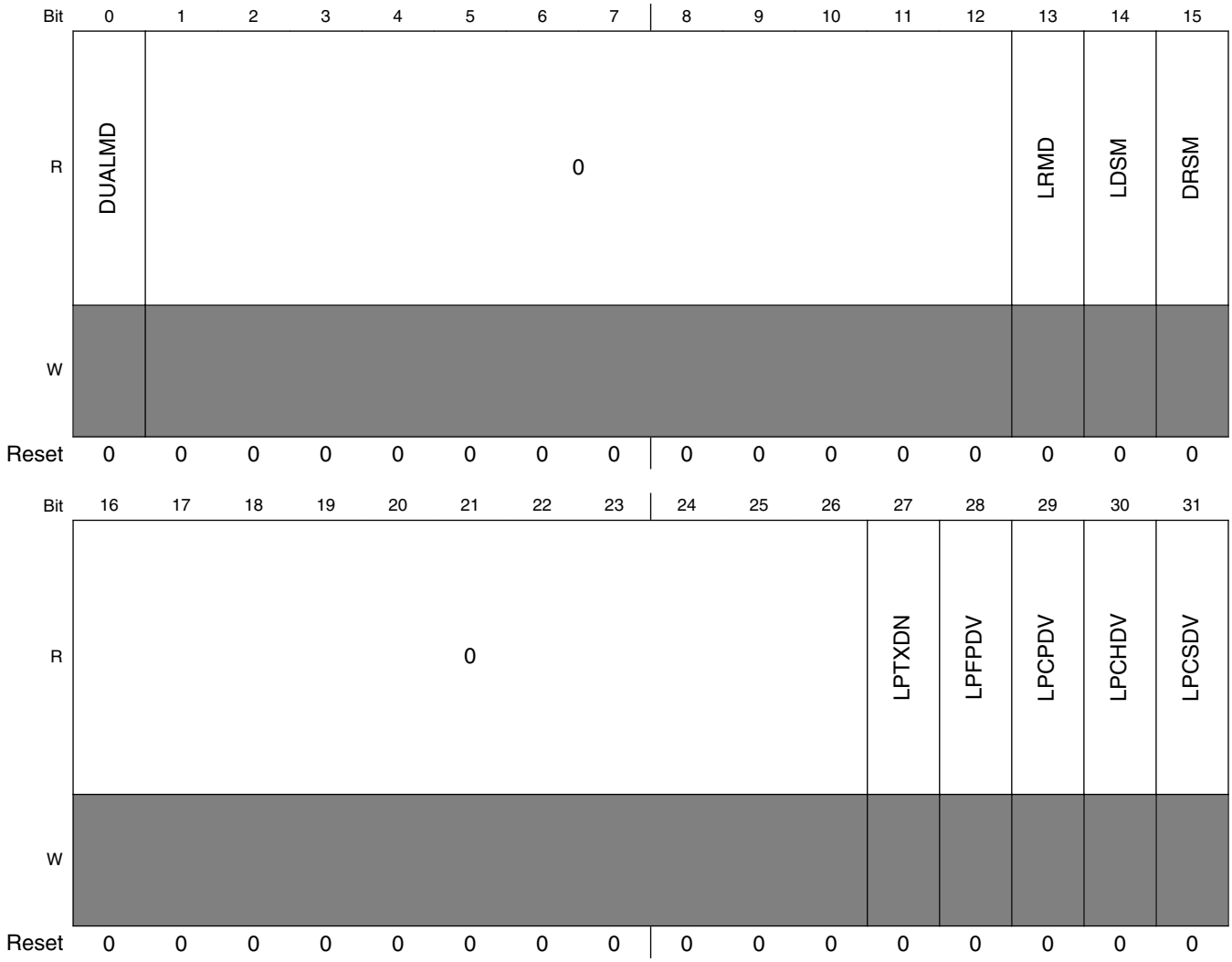
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UNTXD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSTDR*n* field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

76.6.12 LFAST Global Status Register (LFAST_GSR)

Address: 1h base + 80h offset = 81h



LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active.

Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

76.6.13 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 1h base + 94h offset = 95h

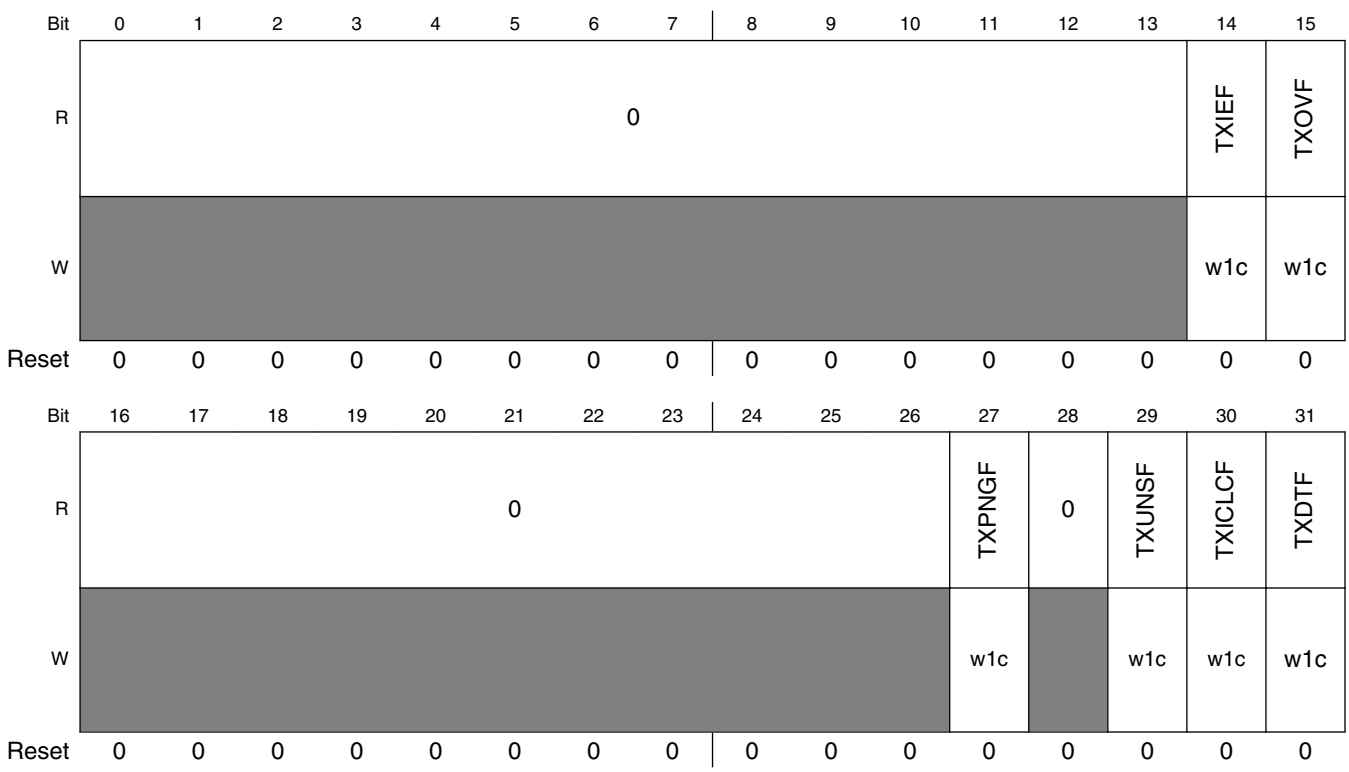
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		RXDCNT						0					RXFCNT			0		TXDCNT							0					TXFCNT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

76.6.14 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 1h base + 98h offset = 99h



LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

76.6.15 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 1h base + 9Ch offset = 9Dh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOF	Reserved		RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c		w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9–10 Reserved	Reserved. The User should ignore the bit in this mode. This field is reserved.
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

76.6.16 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)

Address: 1h base + A0h offset = A1h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received

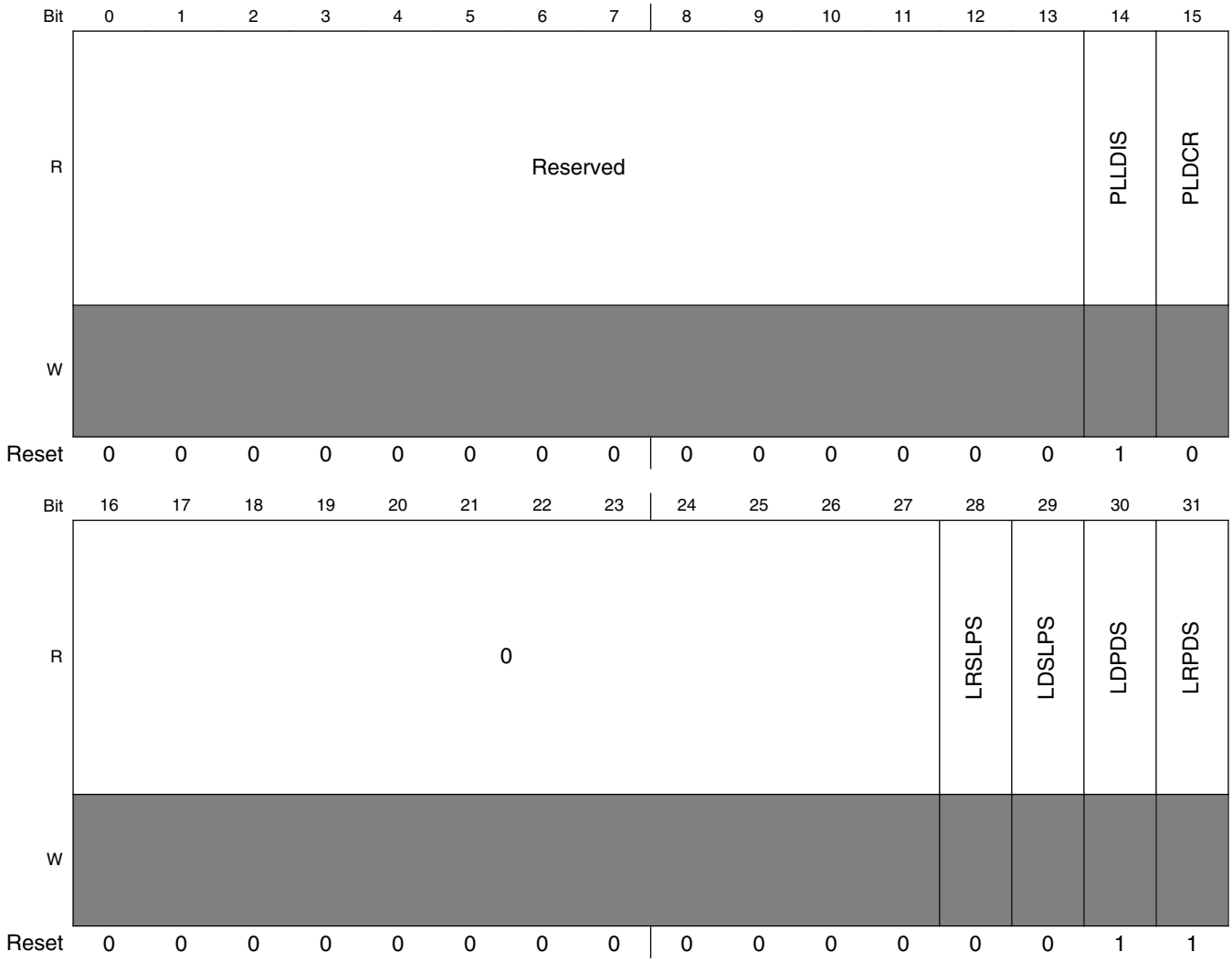
Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
	0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

76.6.17 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 1h base + A4h offset = A5h



LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

76.6.18 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 1h base + A8h offset = A9h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0					URPCNT		
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

76.6.19 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)

Address: 1h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR*n* field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

76.7 Functional description

76.7.1 LFAST Interface enable signal (lfast_sysclk_en)

The LFAST interface enable signal (lfast_sysclk_en) signal is present only in the case LFAST Slave . It also functions as the LFAST Interface control Signal. The interface control functions are:

LFAST interface enable (lfast_sysclk_en) negation:

1. If LCR[SWONLD] = 0, the LD LVDS pad is shut down.
2. If LCR[SWONLR] = 0 the LR LVDS pad is shut down.

3. The LFAST slave exits the test mode (for example, loopback mode or clock test mode). Then S/W writes MCR[CLKTST] = 0 and MCR[LPON] = 0.
4. The LFAST slave Rx and Tx interfaces are set to Low Speed mode. Then write SCR[TDR] = 0 and SCR[RDR] = 0.
5. The Tx and Rx interface controllers are disabled.
6. The status registers are cleared.

LFAST interface enable (lfast_sysclk_en) assertion:

1. The LD LVDS pad is power up, if all conditions are met:
   - LCR[SWOFFLD] = 0.
   - LCR[SWONLD] = 1.
   - MCR[TXEN] = 1.
   - When frame ICR[ICLCPLD] = 31h is received.
2. LR LVDS pad is powered up if all conditions are met:
   - LCR[SWOFFLR] = 0.
   - LCR[SWONLR] = 1.
   - MCR[RXEN] = 1.

76.7.2 Line Receiver

This section describes the Line Receiver.

76.7.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 76-5 shows the block diagram of Uplink Controller.

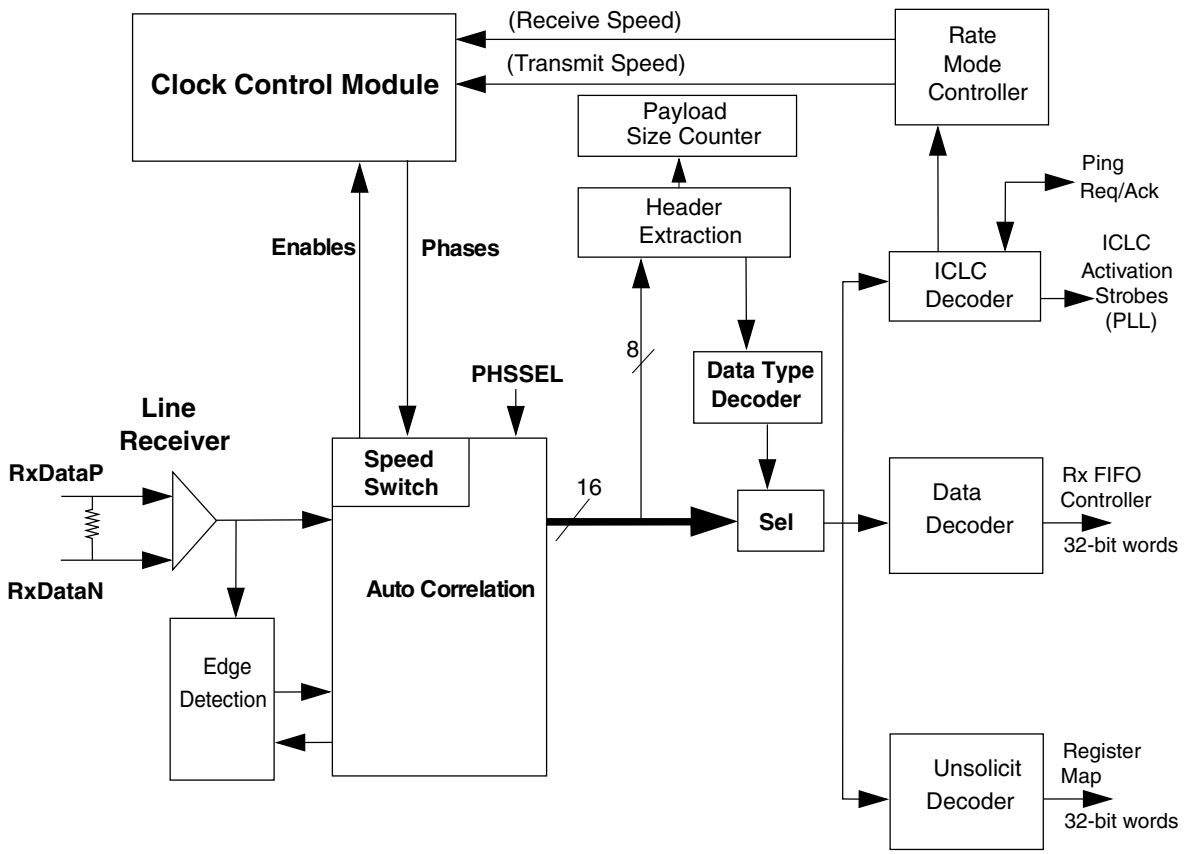


Figure 76-5. Top Level Receive Controller

76.7.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

76.7.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

76.7.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 76-6 shows the block diagram of Hunt Correlation mode.

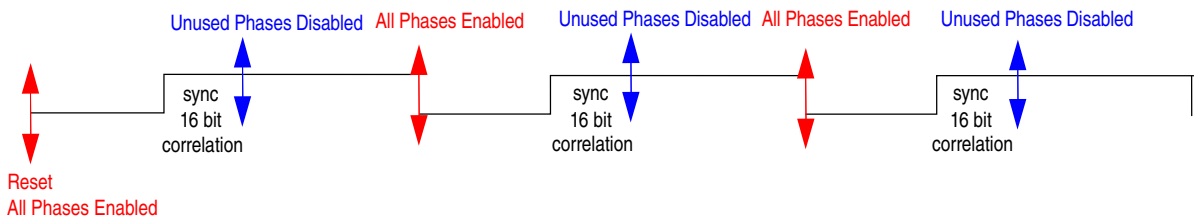


Figure 76-6. Hunt Correlation mode

76.7.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

76.7.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

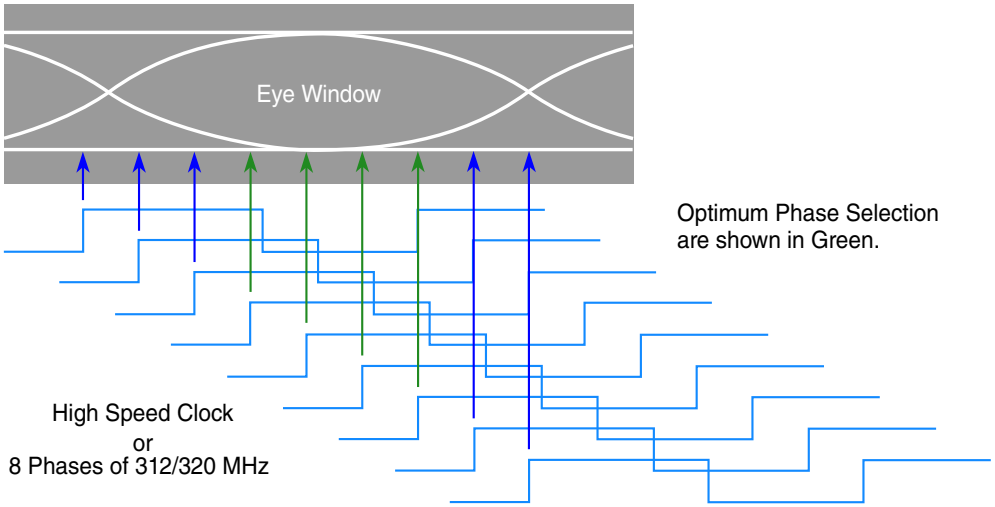


Figure 76-7. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

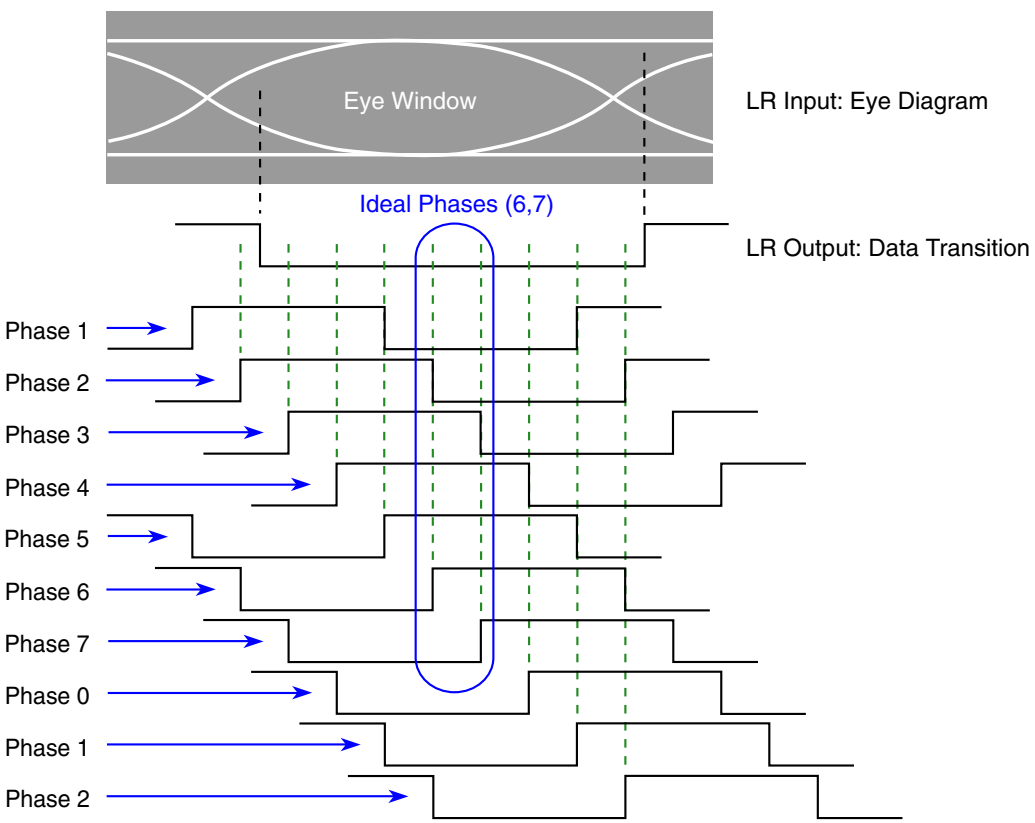


Figure 76-8. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in Figure 76-9 and Figure 76-10.

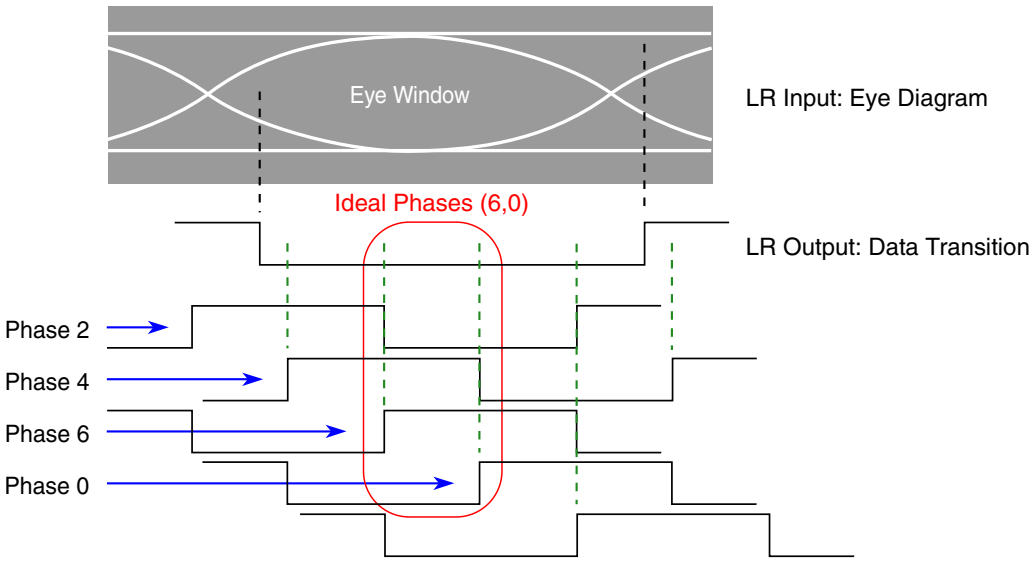


Figure 76-9. High Speed 4 Phase Clock Alignment Example

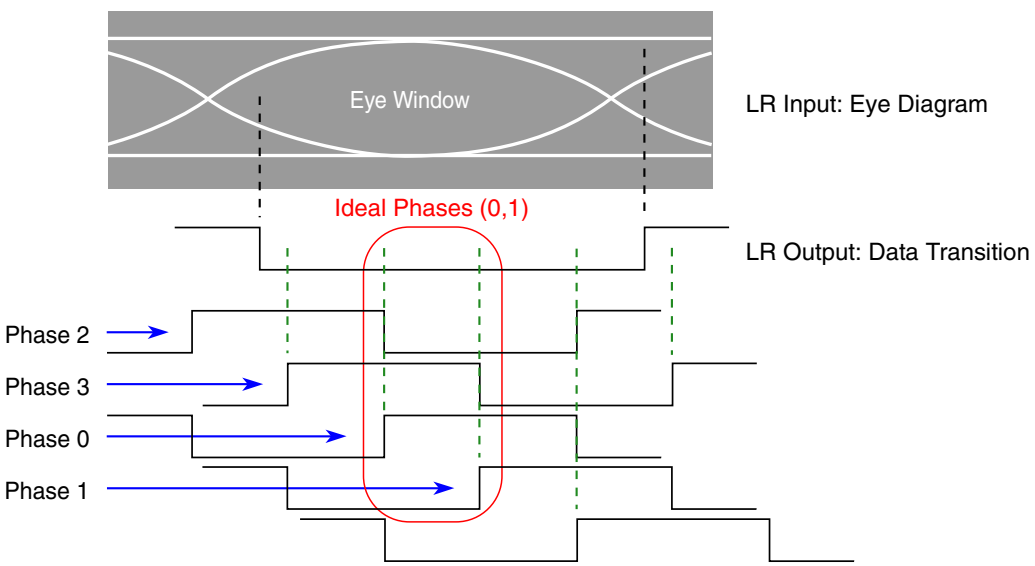


Figure 76-10. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed (312/320 MHz) 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in Table 76-3, Table 76-4, and Table 76-5 before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 76-3. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 76-4. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See Table 76-4, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 76-5. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 76-5. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See Table 76-5, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

76.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

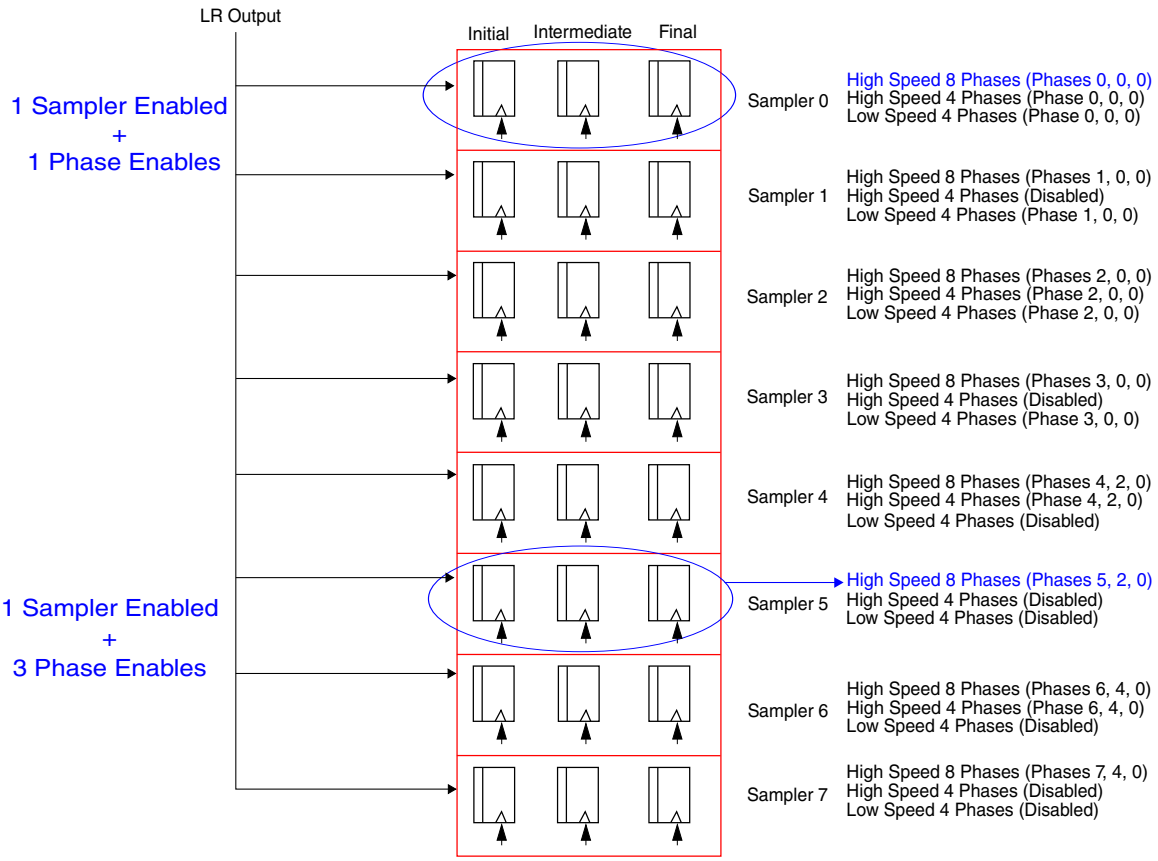


Figure 76-11. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

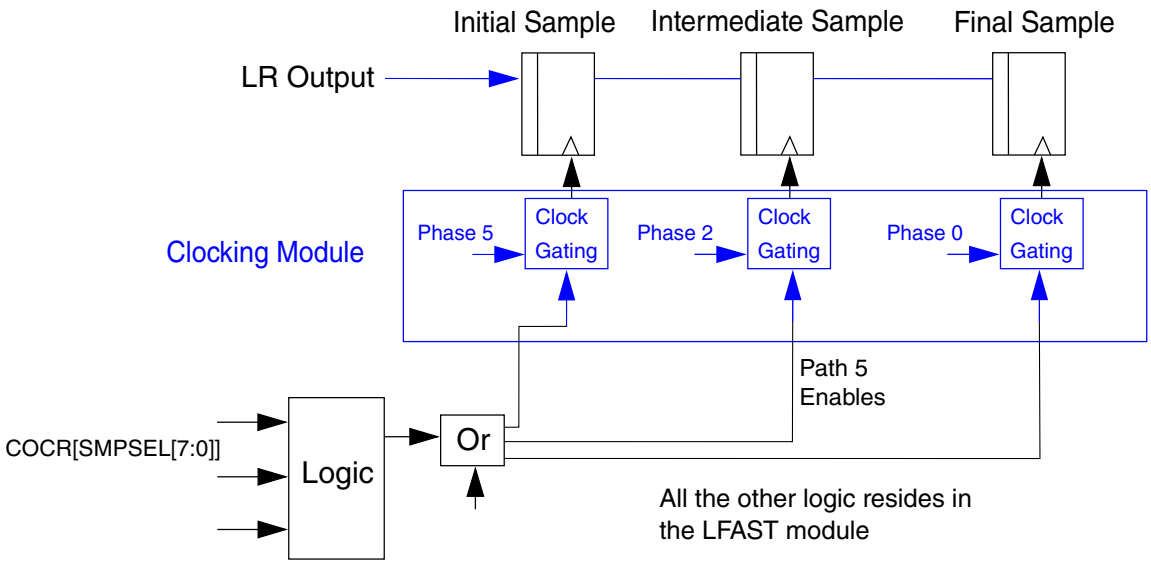


Figure 76-12. Sampler 5 Logic

76.7.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the Figure 76-13.

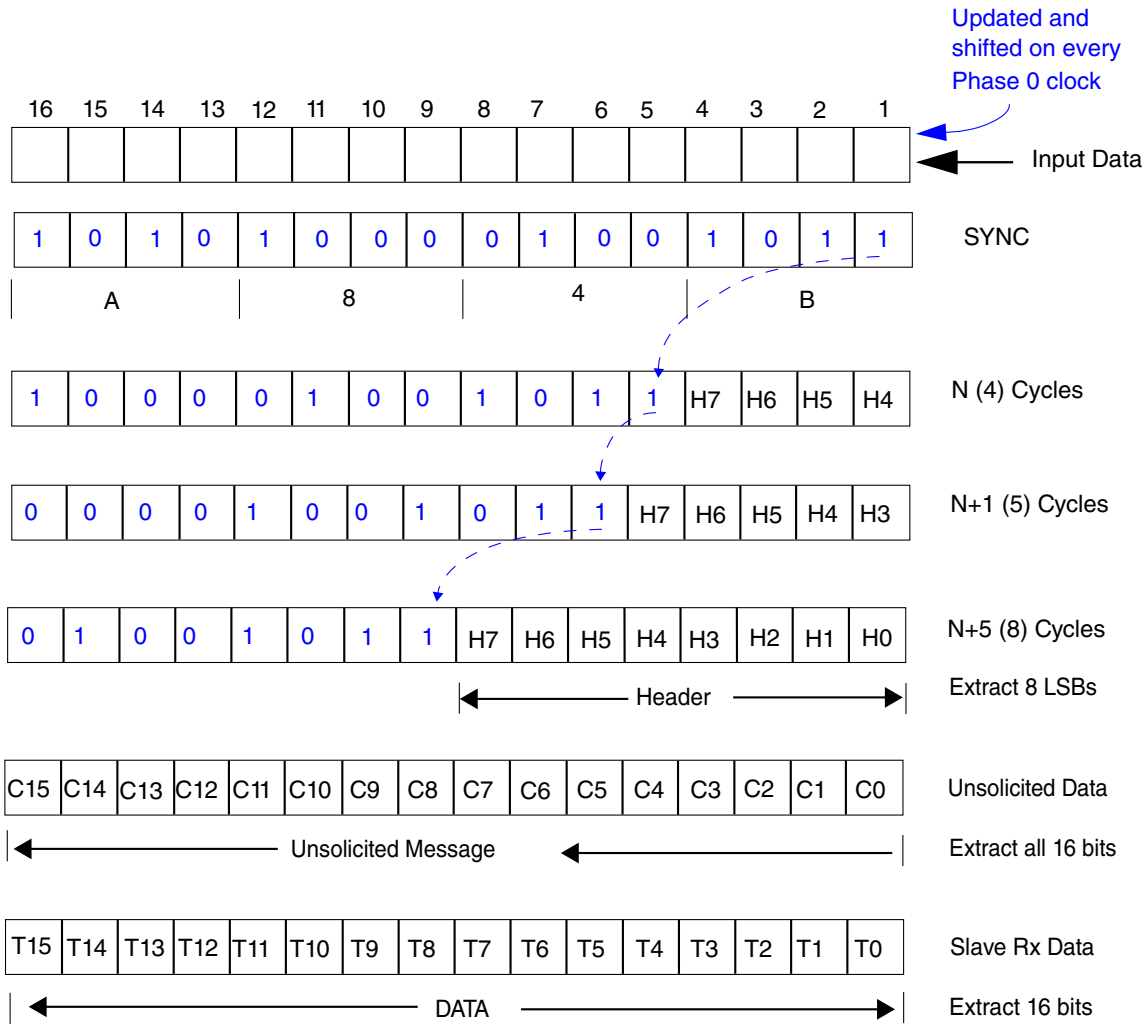


Figure 76-13. Extracting Header and Payload from the 16-bit output from Auto correlation.

76.7.3 Transmit Controller

This section describes the Transmit Controller.

76.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

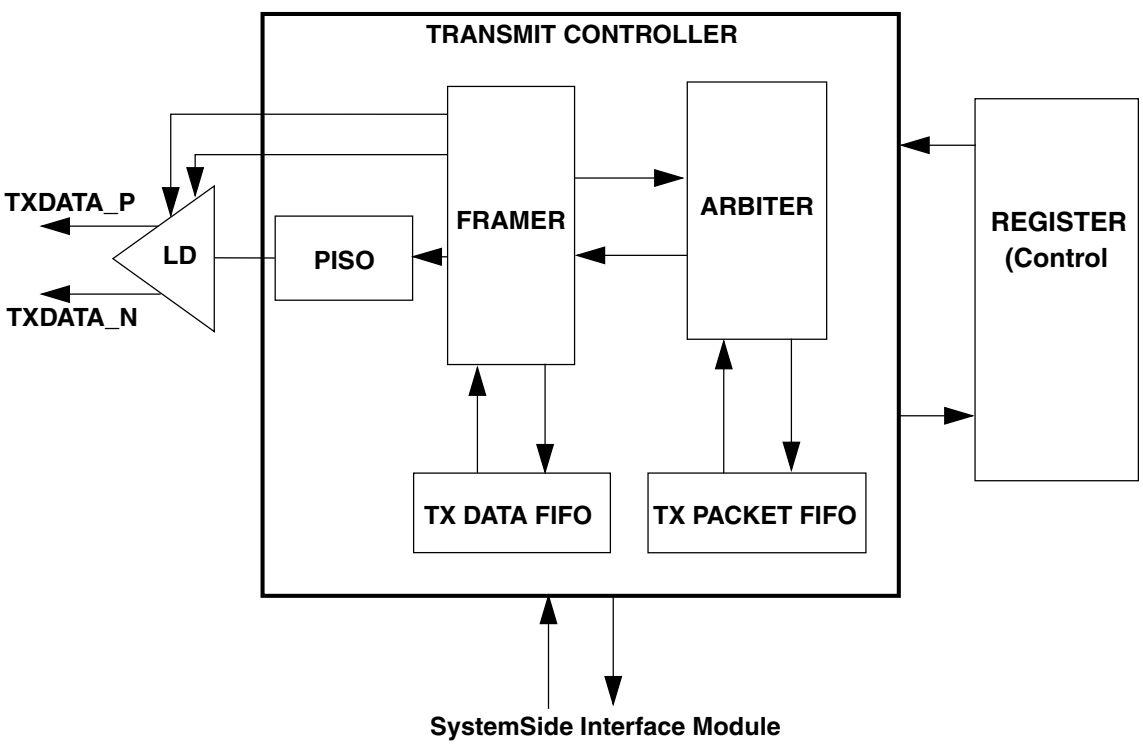


Figure 76-14. Transmit Controller Connections

76.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller
- LFAST Slave: LFAST interface enable (lfast_sysclk_en)

The level of priority for each request is shown in Table 76-6

Table 76-6. Priority Levels for the Tranmsit Controller

Request	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
LFAST interface enable (lfast_sysclk_en)	1	• LFAST interface enable (lfast_sysclk_en) is asserted • LFAST interface enable (lfast_sysclk_en) is negated
Tx Interface disabled	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	• TMCR[LPON] = 1 • LFAST Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
Ping response request	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	• Valid only if Last Frame with header b0 = 1 received from the peer LFAST • UNSTCR[USNDRQ] = 1

Table continues on the next page...

Table 76-6. Priority Levels for the Tranmsit Controller (continued)

Request	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
Data frame from Tx FIFO	7	• Tx FIFO contains one or more frames • Last Frame with header b0 = 1 received from the peer LFAST device • MCR[DATAEN] = 1
Clock mode test	8	• TMCR[CLKTST] = 0 • LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

76.7.3.3 Line Driver digital connections

76.7.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

  The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down

completely. In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

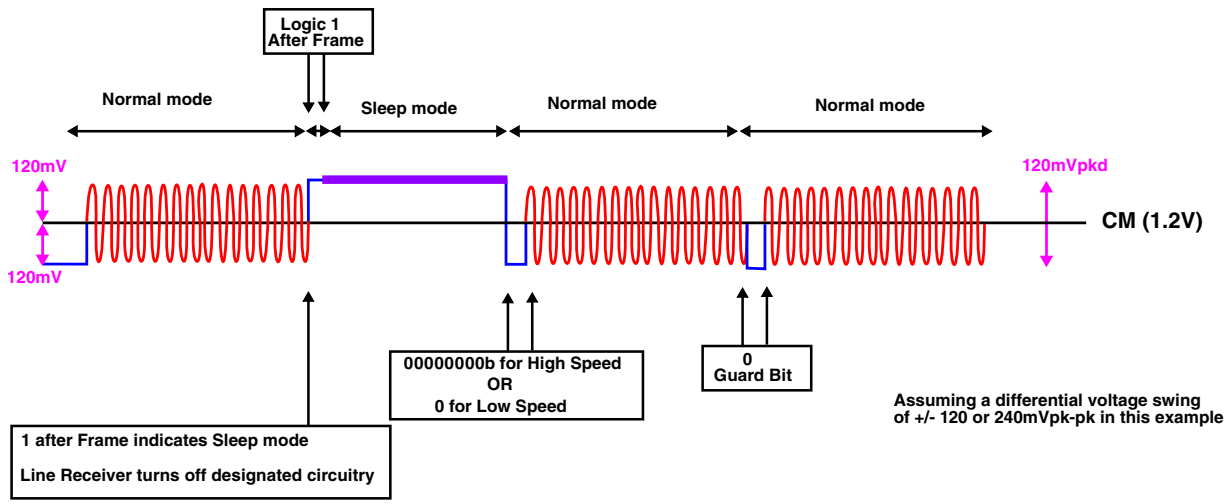


Figure 76-15. Example of Sleep mode

As shown in Figure 76-15 before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 76-7. Line Driver States

LD State	LFAST Slave Triggers
Shutdown	• LFAST interface enable (lfast_sysclk_en) negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions

76.7.4 Frames supported

Table 76-8 provides the frames supported and their permissible payload sizes

Table 76-8. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	Rx – 0100b,1000b – 1011b Tx – All	YES	YES	TX: 96 RX:128
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO[1]	8
CTS Frame	n/a	NO	NO	n/a
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

76.7.5 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

76.7.5.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

76.7.5.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and

schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

76.7.5.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer

76.7.5.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded.

76.7.5.2 Unsolicited flow

76.7.5.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

76.7.5.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.

76.7.5.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.

76.7.5.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

76.7.5.3.1 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

76.7.5.3.1.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

76.7.5.3.1.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

76.7.5.3.1.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

76.7.5.3.1.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

76.7.5.3.1.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

76.7.5.3.1.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

76.7.5.3.1.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable (lfast_sysclk_en).

76.7.6 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

76.7.6.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 76-9. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

76.7.6.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. Bit b0 of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

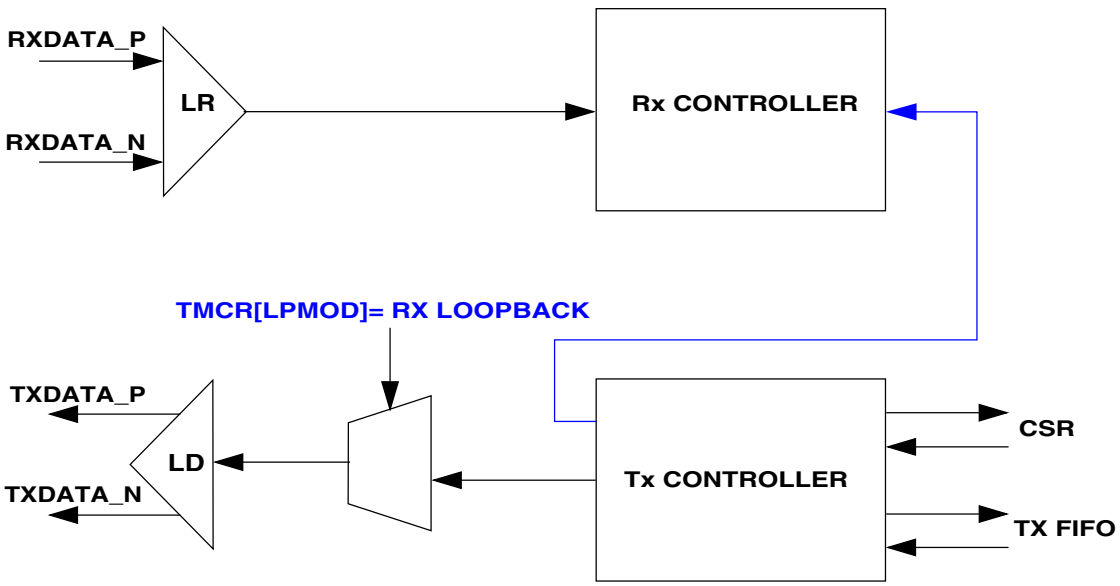


Figure 76-16. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master
- For LFAST Slave: - Deassertion of the LFAST interface enable (lfast_sysclk_en).

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

76.7.6.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

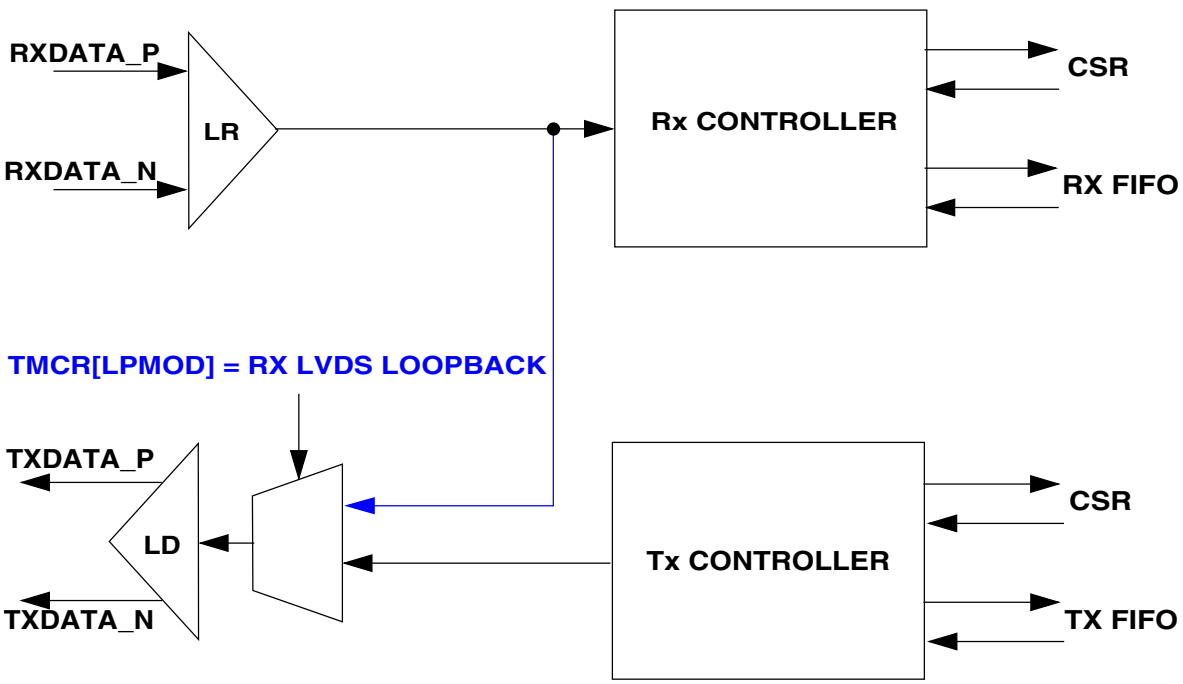


Figure 76-17. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
   - S/W writes TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.
  - Deassertion of the LFAST interface enable (lfast_sysclk_en).

76.7.6.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

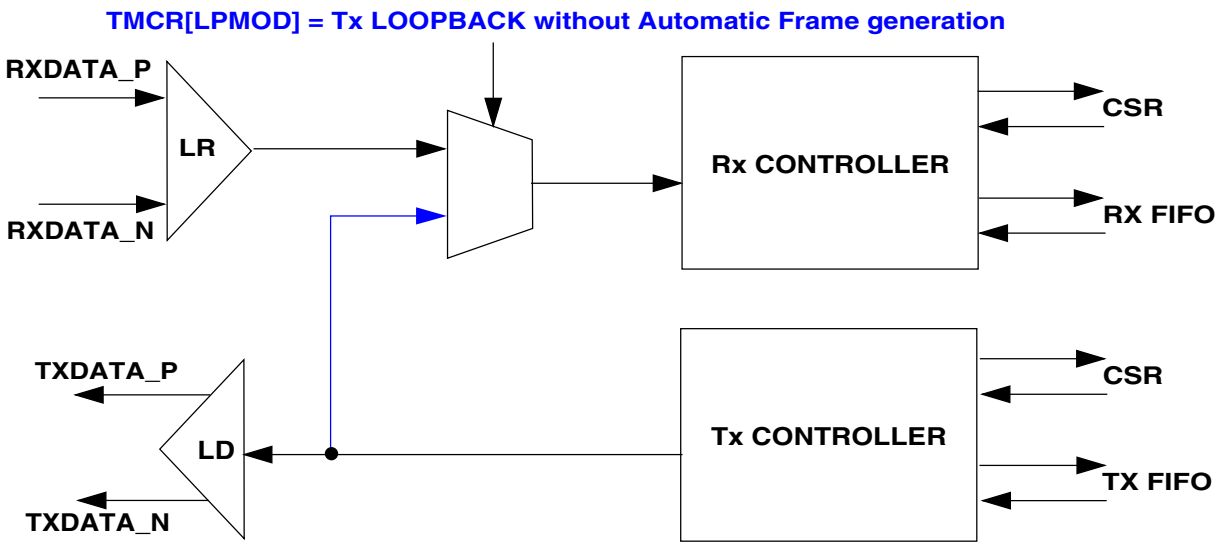


Figure 76-18. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

All frames and error flags are decoded in this mode.

76.7.6.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

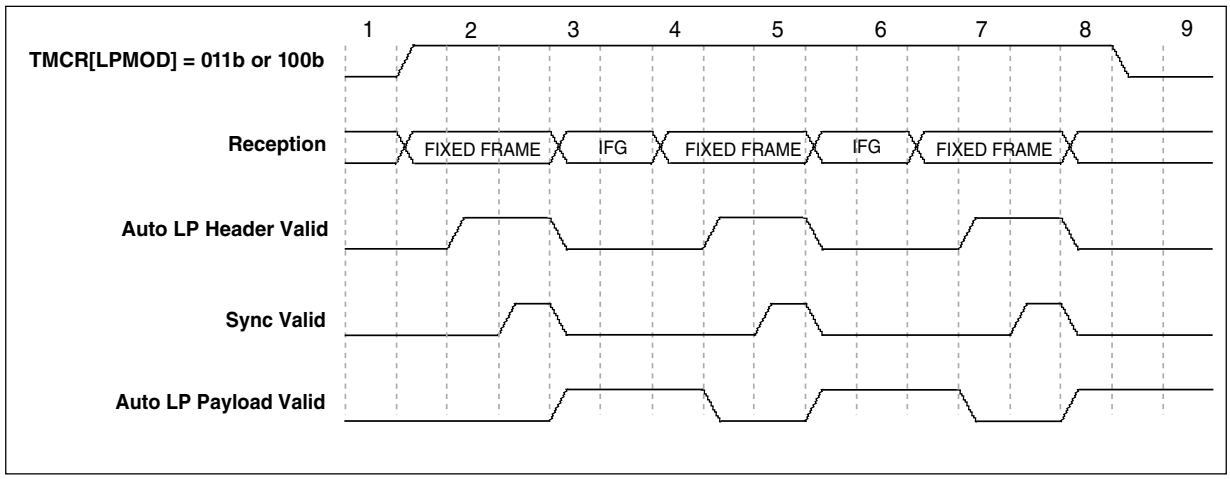


Figure 76-19. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

76.7.6.1.4.1 Tx loopback mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

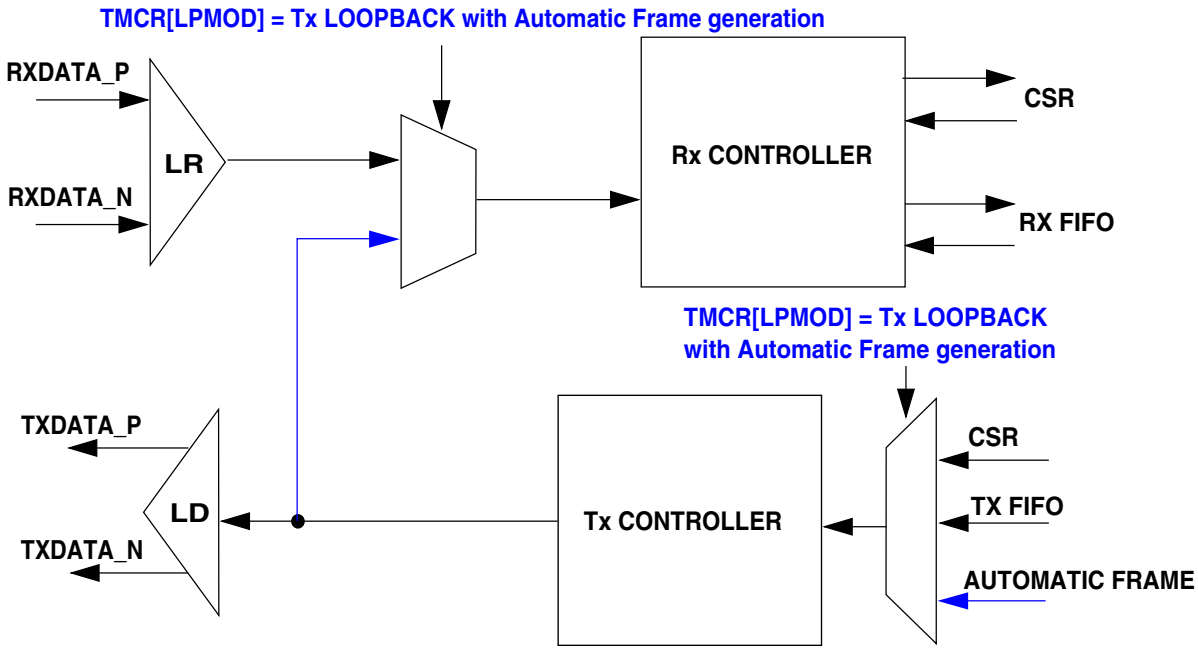


Figure 76-20. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

76.7.6.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

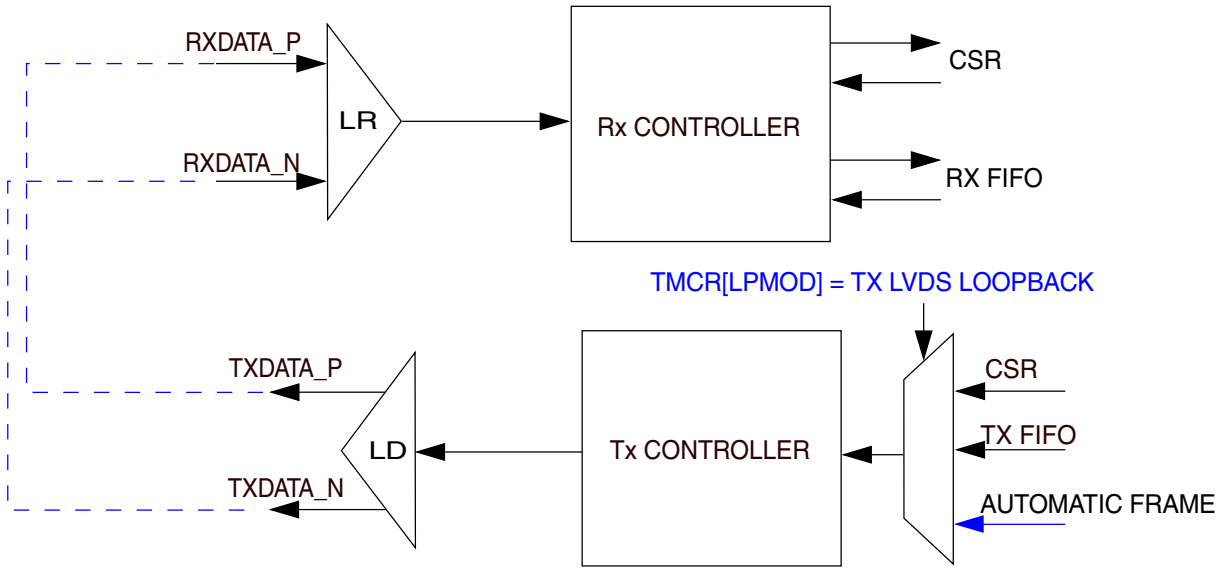


Figure 76-21. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

76.7.6.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 34h (Clock Test mode on) from LFAST Master

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

76.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 76-10. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	6 × 32 max 2 packets	8 × 32 max 2 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	N/A	1 × 8 max 1 packet[3]	Registers PICR[PNGPYLD]
CTS Frame	N/A	N/A	N/A

3. Ping response packet

76.9 Resets

The various blocks in LFAST are reset as described in the table below.

Table 76-11. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]
LFAST interface enable (lfast_sysclk_en)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers except the following: TISR, RISR, RIISR, GSR[LPTXDN] and GSR[LPFPDV] These bits remain reseted while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: Negedge only • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON]

Table continues on the next page...

Table 76-11. Recommended receive exception handling mechanism (continued)

Reset	Blocks Reset
	• ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ] These bits doesn't remain reseted while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: positive edge only • TISR, • RISR, • RIISR, • GSR[LPTXDN] and GSR[LPFPDV] • These bits doesn't remain reseted while the LFAST interface enable (lfast_sysclk_en) is High.

76.10 Clocks

The LFAST mainly works on two clocks:

- System Bus Clock used to program the registers.
- System Side Module clock, which is synchronous to the System Bus clock.

76.10.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

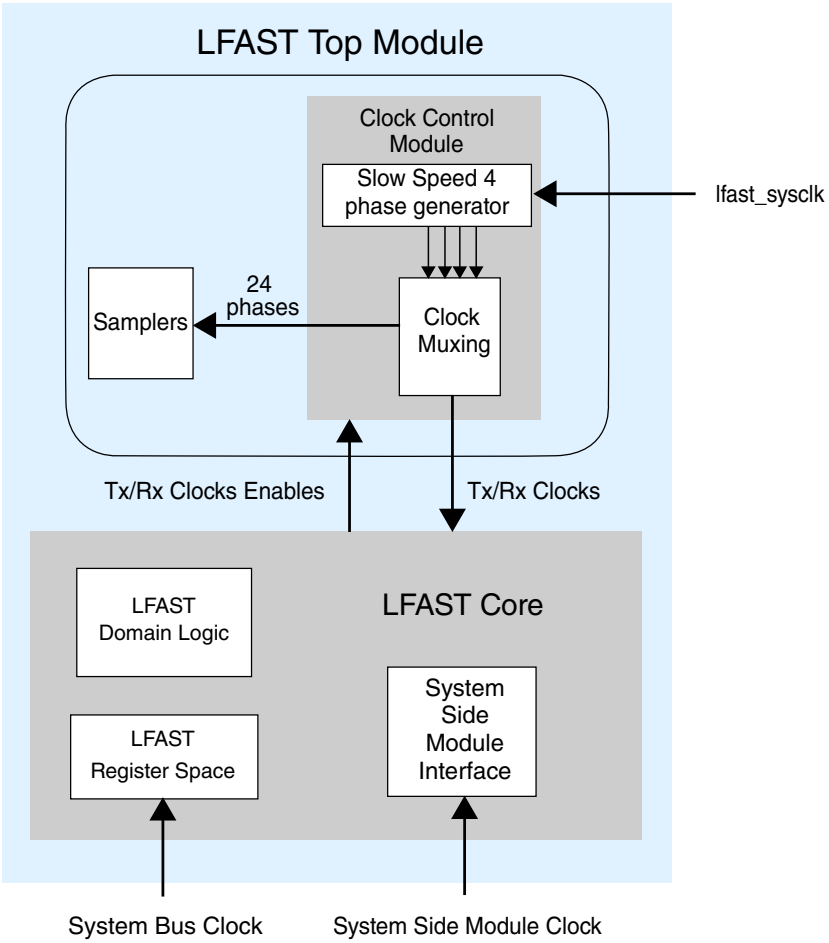


Figure 76-22. LFAST module clock domains

76.10.2 Slow speed clock

76.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in Figure 76-23, Figure 76-24 and Figure 76-25.

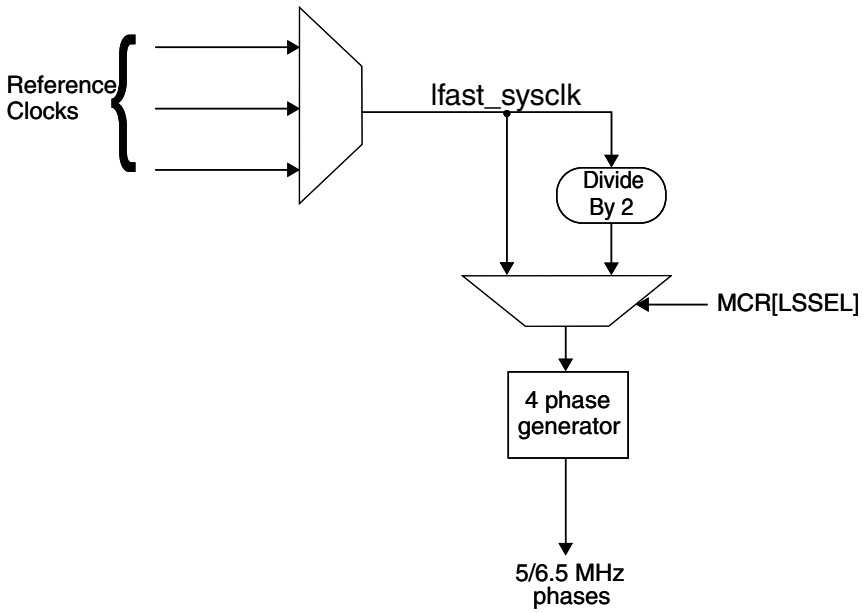


Figure 76-23. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in Figure 76-23.

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 76-24 (selected clock path shown in green).

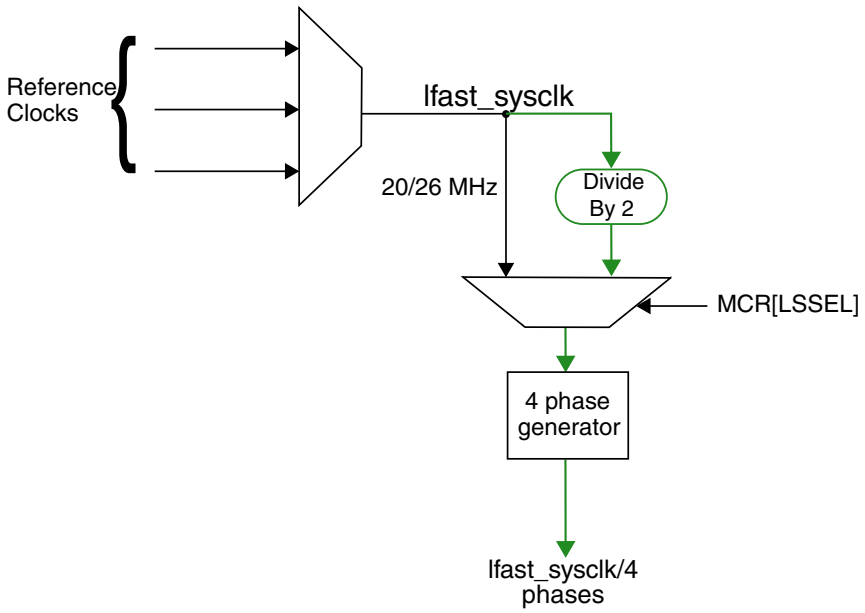


Figure 76-24. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 76-25 (selected clock path selected shown in magenta).

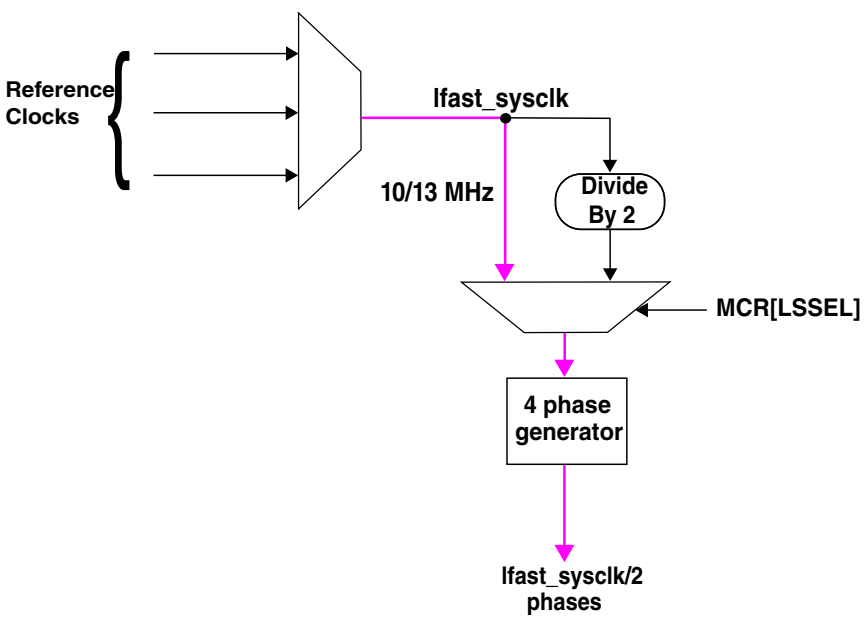


Figure 76-25. External clock muxing of lfast_sysclk of 10 MHz

76.10.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

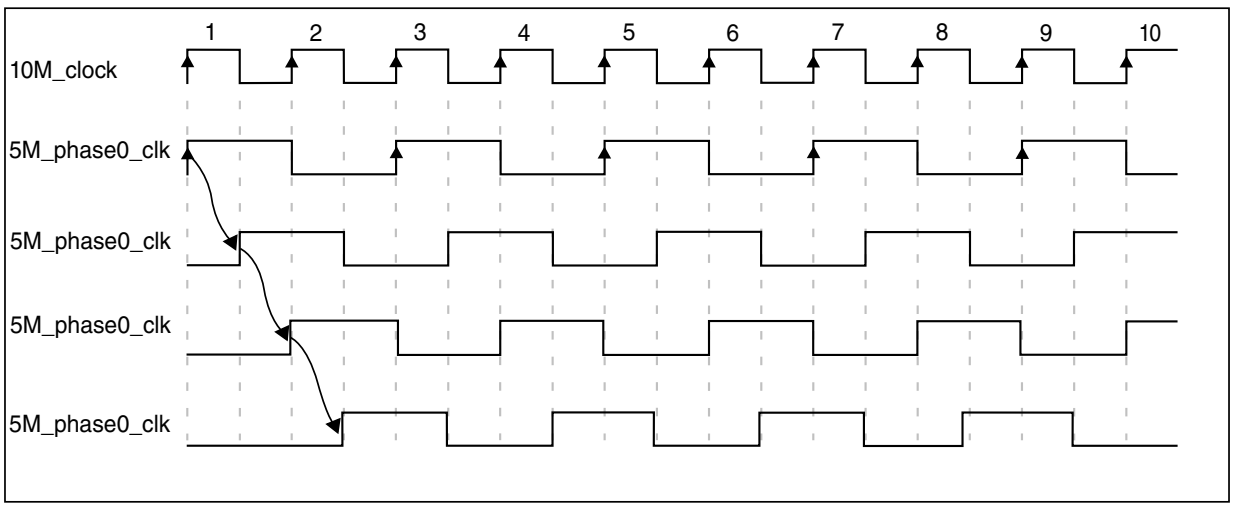


Figure 76-26. Slow speed 4 phases generation

76.10.3 Rx Controller Clocks

76.10.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

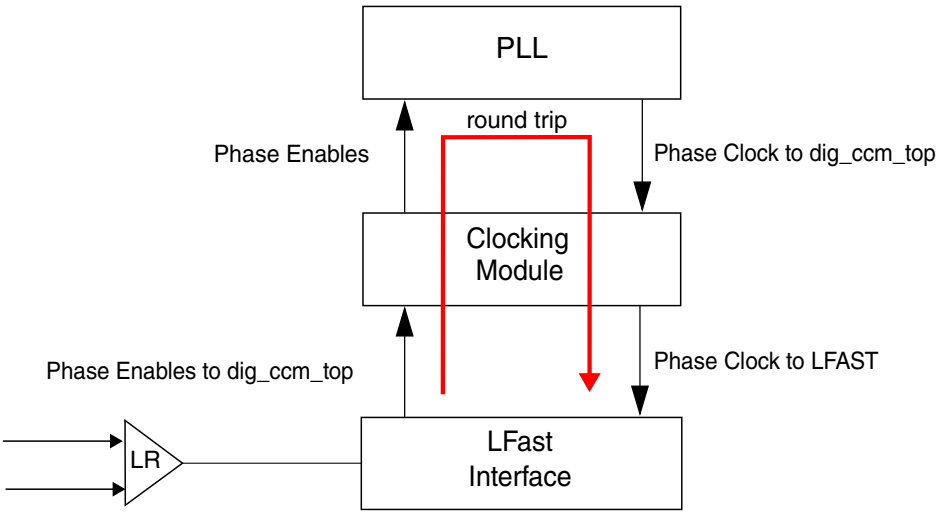


Figure 76-27. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

76.10.5 Clock module requirements for low speed phases

76.10.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 76-12. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

76.10.6 Tx Controller Clocks

76.10.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 76-13. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

76.10.6.2 Transmit Clock Muxing

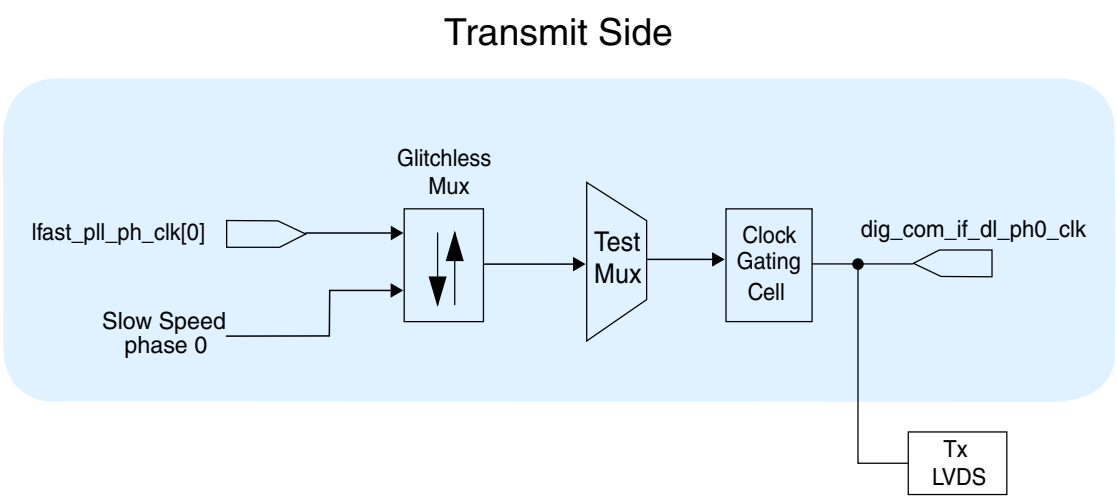


Figure 76-28. Transmit Clocks Muxing

76.10.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

Chapter 77
Development Trigger Semaphore (DTS)

77.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note

When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2012 Nexus standards, which use the Nexus Auxiliary port.

77.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal—DTS Trigger Output (DTO). The registers are as follows:

- DTS_SEMAPHORE register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access or debug Zipwire.

- DTS_SEMAPHORE_B register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN_B bit is set in the DTS_ENABLE register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access or debug Zipwire.
- DTS_STARTUP register—This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register—This register provides an enable/disable capability for the DTS feature.

The architecture is shown in the following figure.

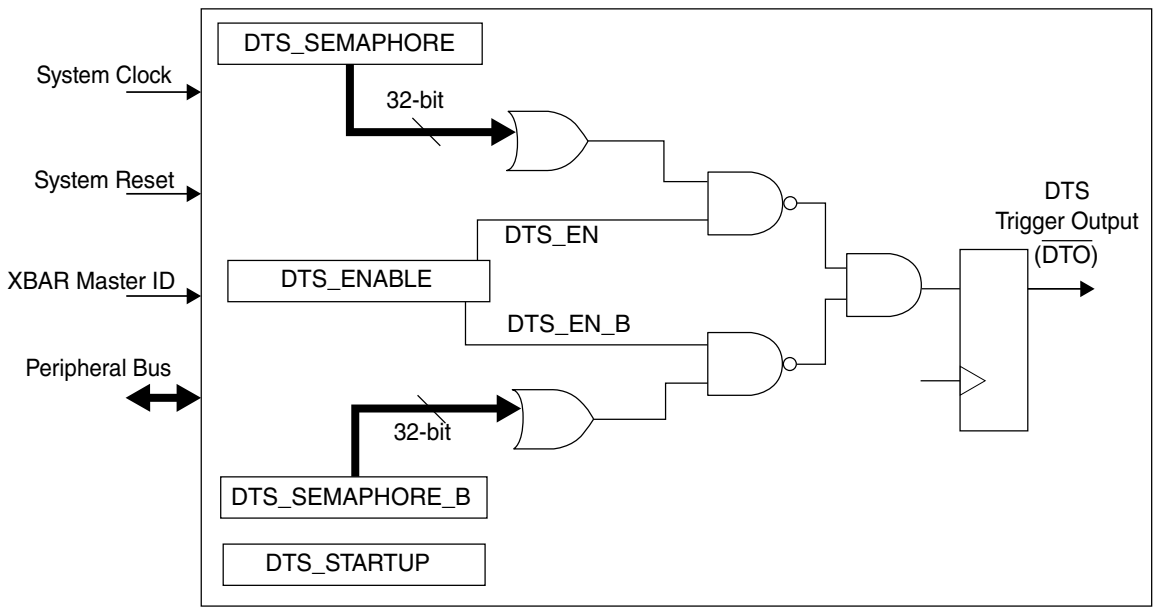


Figure 77-1. DTS block diagram

The DTS Trigger Output (DTO) signal is connected to one of the EVTO inputs of the Debug and Calibration Interface (DCI). The other EVTO inputs are connected to the other Nexus modules in the device. DTO is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note

When the DTS module is enabled (DTS_ENABLE[DTS_EN] = 0b1 or DTS_ENABLE[DTS_EN_B] = 0b1), all other Nexus EVTO functions associated should be disabled by the tool and

EVTO becomes the DTO. Unlike the EVTO function that only asserts for one clock, the DTO function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

The following figure shows an example of the chain of events that begins with setting of a bit in the DTS_SEMPAHORE register and the clearing of the register caused by a Nexus read.

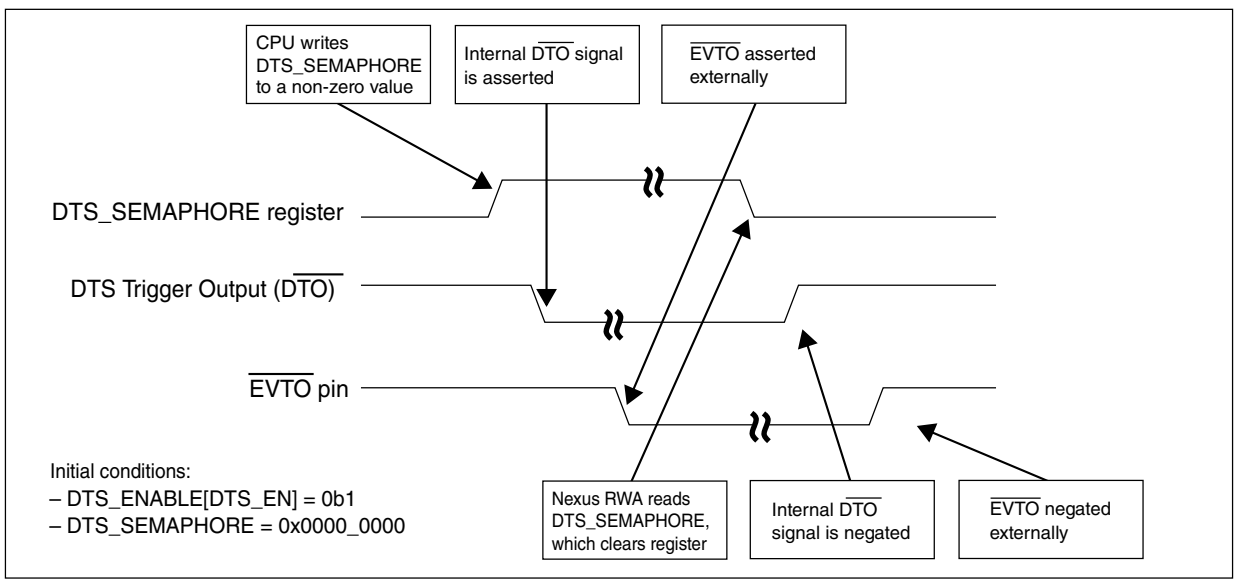


Figure 77-2. DTO event sequence example

77.3 DTS device connections

The following figure shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

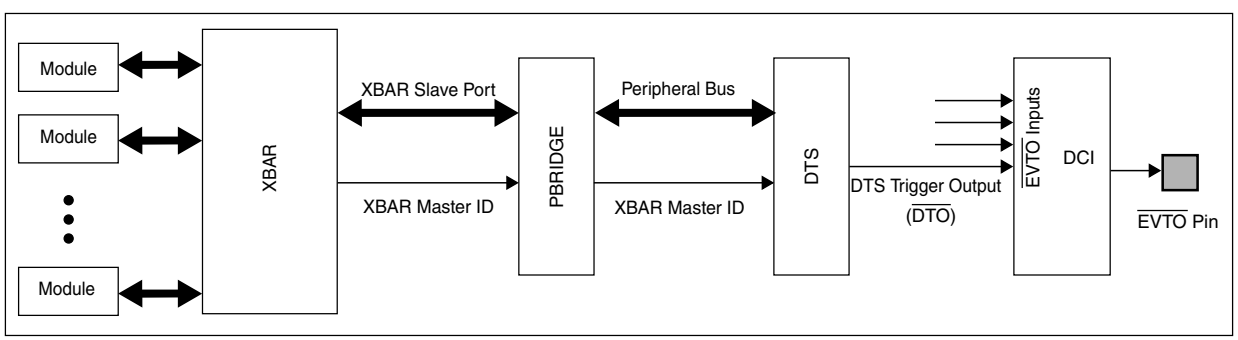


Figure 77-3. DTS device connections

The registers have limited access as described in DTS register access. Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool or debug Zipwire can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access or debug Zipwire.

Note

Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

77.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in the following table. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 77-1. DTS register access effects

Register	32-bit read				32-bit write			
	RWA[1] or debug Zipwire	Core	eDMA	Other crossbar masters	RWA [1] or debug Zipwire	Core	eDMA	Other crossbar masters
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear[2]	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear[2]	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool
2. A read of the semaphore registers by Nexus Read/Write Access module or debug Zipwire is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note

The XBAR Master ID should not be confused with the Master Port number of the XBAR. See the Crossbar Switch (XBAR) chapter for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core or debug Zipwire. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

77.4 Memory mapped registers

Only certain types of accesses are allowed to the DTS registers. See the description for each register to determine the accesses that are allowed.

DTS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Output Enable Register (DTS_ENABLE)	32	R/W	0000_0000h	77.4.1/4077
4	Startup Register (DTS_STARTUP)	32	R/W	0000_0000h	77.4.2/4078
8	Semaphore Register (DTS_SEMAPHORE)	32	R/W	FFFF_FFFFh	77.4.3/4079
C	Semaphore Extension (DTS_SEMAPHORE_B)	32	R/W	FFFF_FFFFh	77.4.4/4080

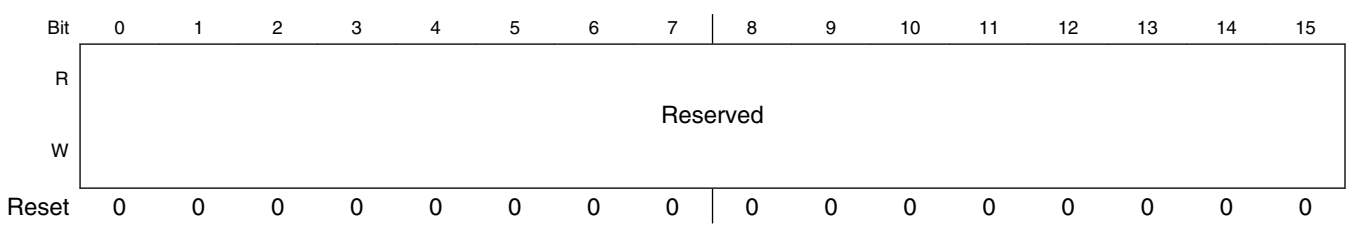
77.4.1 Output Enable Register (DTS_ENABLE)

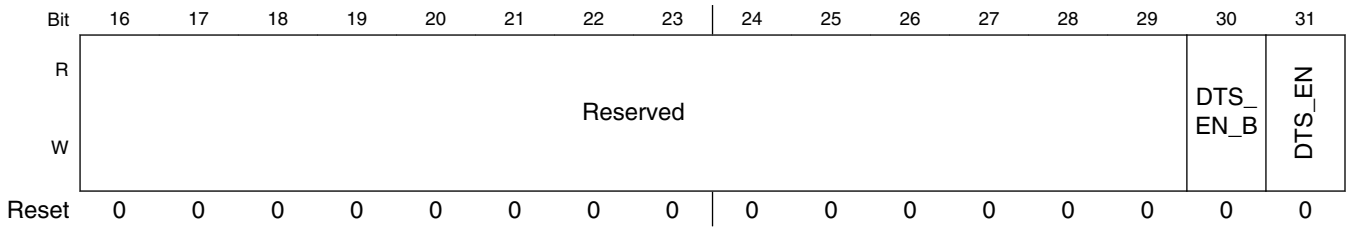
The DTS_ENABLE register controls the DTS Trigger Output ($\overline{\text{DTO}}$) and whether $\overline{\text{DTO}}$ is active on the $\overline{\text{EVTO}}$ output pin of the device. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA) or debug Zipwire.

NOTE

Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

Address: 0h base + 0h offset = 0h





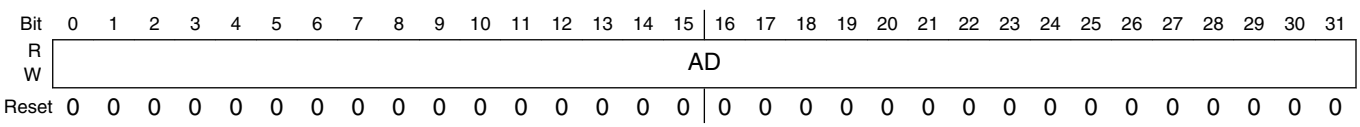
DTS_ENABLE field descriptions

Field	Description
0–29 Reserved	This field is reserved.
30 DTS_EN_B	DTS Enable B. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).
31 DTS_EN	DTS Enable. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).

77.4.2 Startup Register (DTS_STARTUP)

The DTS_STARTUP register is used for tool detection and startup information exchange between external data acquisition tool and the embedded controller. DTS_STARTUP register can be read by the core, the eDMA module and Nexus but can only be updated by a Nexus Read Write Access (RWA) or debug Zipwire.

Address: 0h base + 4h offset = 4h



DTS_STARTUP field descriptions

Field	Description
0–31 AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application

DTS_STARTUP field descriptions (continued)

Field	Description
	software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access or debug Zipwire to update the contents of this register. A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

77.4.3 Semaphore Register (DTS_SEMAPHORE)

The DTS_SEMAPHORE register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DTS_SEMAPHORE field descriptions

Field	Description
0–31 ST	Semaphore Trigger. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. • All register bits are set to '1' by a device reset. • A Nexus reset does not change the state of this register. • The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA or debug Zipwire. • For the core or DMA, only 32-bit write or read accesses are valid. • A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

77.4.4 Semaphore Extension (DTS_SEMAPHORE_B)

The DTS_SEMAPHORE_B register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST_B																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DTS_SEMAPHORE_B field descriptions

Field	Description
0–31 ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. • All register bits are set to '1' by a device reset. • A Nexus reset does not change the state of this register. • The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA or debug Zipwire. • For the core or DMA, only 32-bit write or read accesses are valid. • A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

77.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note

It is the user's responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

The following table shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA—all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

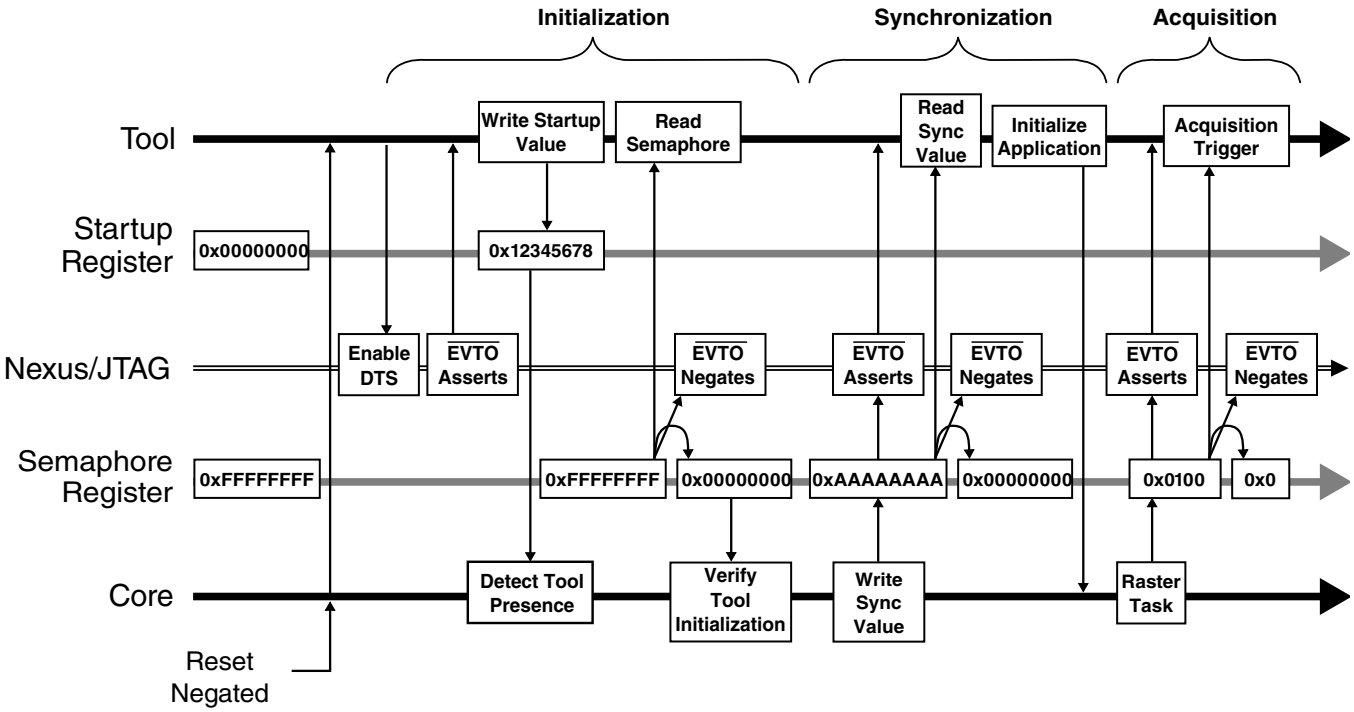


Figure 77-4. DTS startup sequence example

Chapter 78
Nexus Aurora Router (NAR)

78.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in the following figure. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured.

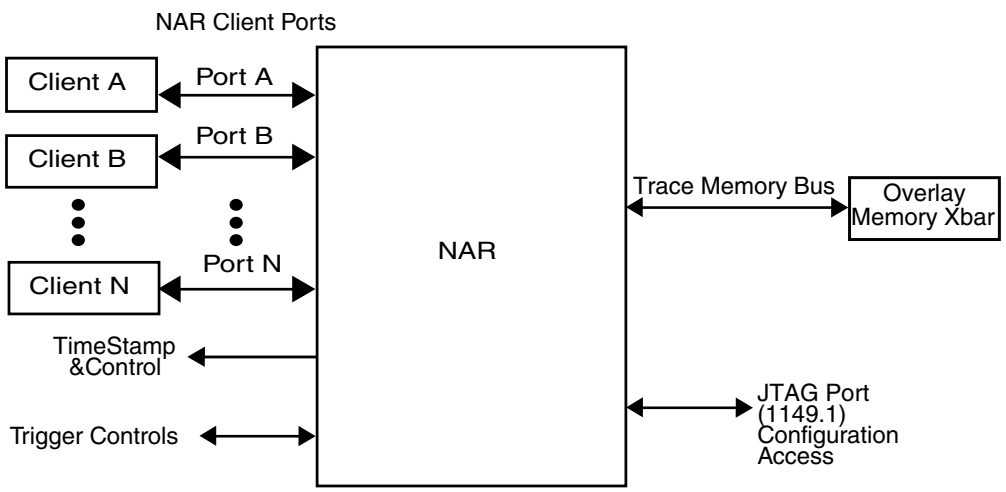


Figure 78-1. NAR top level block diagram

The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. The Production Device (PD) does not support partitioning of the main queue; therefore, only one of the ports can be active at a time. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The NAR supports the following client interfaces:

- Legacy interface to support existing client interface protocol through a conversion gasket logic with clock domain transfer
- New client interface with clock domain transfer

78.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (IEEE 1149.1) port.

78.3 Features

The NAR supports the following features:

- Connects 7 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Legacy client interface for backward compatibility

- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Time-stamping provides a central synchronization resource for message ordering. Gray coded timer output for clients working in different clock domians.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

78.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (IEEE 1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 78-1. NAR registers

Register name	Read/Write	JTAG index (NAR relative)
NAR_CR—NAR Control Register	R/W	0
NAR_ST—NAR Status Register	R	1
NAR_SFR—Source Filter Register	R/W	2
NAR_TFR—Type Filter Register	R/W	3
NAR_TBALO—Trace memory port base address, low	R/W	4
NAR_TBAHI—Trace memory port base address, high	R/W	5
NAR_TCR—Trace Memory bus control register	R/W	6

Table continues on the next page...

Table 78-1. NAR registers (continued)

Register name	Read/Write	JTAG index (NAR relative)
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)
NAR_CSSR–Client Suppress Status Register	R	B
NAR_CDR—Client Disable Register	R/W	C
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E

78.4.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following table shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	0	0	0	ALT		0	AFA	0	0	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CR is described in the following table.

Table 78-2. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See Disabled for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages

Table continues on the next page...

Table 78-2. NAR_CR field descriptions (continued)

Field	Description
	1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). **Note:** Some clients may only support timestamps on every message. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 Reserved 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.
10–9	Reserved and must be written with 0s.
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 Aurora out is the ALT 10 Trace memory bus is the ALT 11 Reserved
5	Reserved

Table continues on the next page...

Table 78-2. NAR_CR field descriptions (continued)

Field	Description
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2	Reserved and must be written with 0.
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

78.4.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following table shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The NAR_ST register is described in the following table.

Table 78-3. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode

Table continues on the next page...

Table 78-3. NAR_ST field descriptions (continued)

Field	Description
	001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode
27–24 NCE	NAR Config Error. See Table 78-17 for coding
23–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer (last valid write + 1)

78.4.3 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results.

78.4.3.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFI	FE0	SF0				0	0	0	FE1	SF1				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	SF2				0	0	0	FE3	SF3				0	ACC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 78-4. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field.
17–15	Reserved

Table continues on the next page...

Table 78-4. NAR_SFR field descriptions (continued)

Field	Description
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

78.4.3.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TFR is described in the following table.

Table 78-5. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field.

78.4.4 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

78.4.4.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following tables show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBAHI																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register index: 4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBALO																						ADRLO[9]	ADRLO[8]	ADRLO[7]	ADRLO[6]	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 78-6. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 78-7. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

78.4.4.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following table shows the format of the NAR_TCR.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTM	BTS		BTH		0	0	0	TBW	MXFR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MXFR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TCR is described in the following table.

Table 78-8. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
23 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

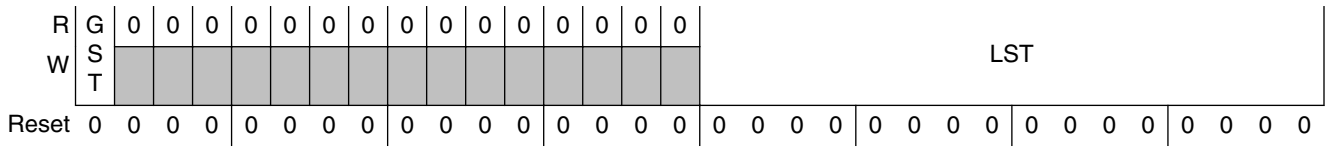
78.4.5 Suppress mode triggers registers (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in Suppress mode. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following table, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7-A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...



The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 78-9. NAR_STCR field descriptions

Field	Description	
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode	
29–16	Reserved	
15–0 LST	Local Suppress Trigger.	
	0000000000000000	The associated trigger is disabled for local suppress mode for all clients
	xxxxxxxxxxxxxxx1	Trigger is enabled for Client 1 local suppress
	xxxxxxxxxxxxxx1x	Trigger is enabled for Client 2 local suppress
	xxxxxxxxxxxxx1xx	Trigger is enabled for Client 3 local suppress
	xxxxxxxxxxxx1xxx	Trigger is enabled for Client 4 local suppress
	xxxxxxxxxxx1xxxx	Trigger is enabled for Client 5 local suppress
	xxxxxxxxxx1xxxxx	Trigger is enabled for Client 6 local suppress
	xxxxxxxxx1xxxxxx	Trigger is enabled for Client 7 local suppress
	xxxxxxxx1xxxxxxx	Trigger is enabled for Client 8 local suppress
	xxxxxxx1xxxxxxxx	Trigger is enabled for Client 9 local suppress
	xxxxxx1xxxxxxxxx	Trigger is enabled for Client 10 local suppress
	xxxxx1xxxxxxxxxx	Trigger is enabled for Client 11 local suppress
	xxxx1xxxxxxxxxxx	Trigger is enabled for Client 12 local suppress
	xxx1xxxxxxxxxxxx	Trigger is enabled for Client 13 local suppress
	xx1xxxxxxxxxxxxx	Trigger is enabled for Client 14 local suppress
	x1xxxxxxxxxxxxxx	Trigger is enabled for Client 15 local suppress
	1xxxxxxxxxxxxxxx	Trigger is enabled for Client 16 local suppress
	All other options	Reserved

78.4.6 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger

causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following table, can then be read to determine which clients are suppressed, and which are active.

Register index: B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

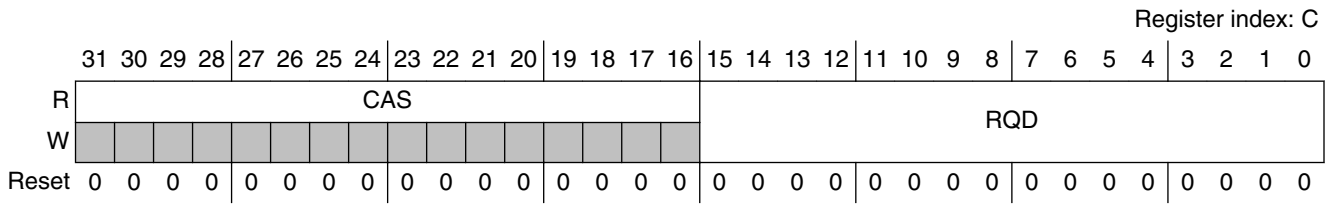
The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 78-10. NAR_CSSR field descriptions

Field	Description	
31–16	Local Suppress Status.	
LSS	0000000000000000	No client is in local suppress mode
	xxxxxxxxxxxxxxx1	Client 1 is suppressed
	xxxxxxxxxxxxxx1x	Client 2 is suppressed
	xxxxxxxxxxxxx1xx	Client 3 is suppressed
	xxxxxxxxxxxx1xxx	Client 4 is suppressed
	xxxxxxxxxxx1xxxx	Client 5 is suppressed
	xxxxxxxxxx1xxxxx	Client 6 is suppressed
	xxxxxxxxx1xxxxxx	Client 7 is suppressed
	xxxxxxxx1xxxxxxx	Client 8 is suppressed
	xxxxxxx1xxxxxxxx	Client 9 is suppressed
	xxxxxx1xxxxxxxxx	Client 10 is suppressed
	xxxxx1xxxxxxxxxx	Client 11 is suppressed
	xxxx1xxxxxxxxxxx	Client 12 is suppressed
	xxx1xxxxxxxxxxxx	Client 13 is suppressed
	xx1xxxxxxxxxxxxx	Client 14 is suppressed
	x1xxxxxxxxxxxxxx	Client 15 is suppressed
	1xxxxxxxxxxxxxxx	Client 16 is suppressed
	All other options	Reserved
15–0	Reserved	

78.4.7 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following table, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.



The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 78-11. NAR_CDR field descriptions

Field	Description	
31–16	Client Activity Status.	
CAS	xxxxxxxxxxxxxxx1	Client 1 is active with new data
	xxxxxxxxxxxxxx1x	Client 2 is active with new data
	xxxxxxxxxxxxx1xx	Client 3 is active with new data
	xxxxxxxxxxxx1xxx	Client 4 is active with new data
	xxxxxxxxxxx1xxxx	Client 5 is active with new data
	xxxxxxxxxx1xxxxx	Client 6 is active with new data
	xxxxxxxxx1xxxxxx	Client 7 is active with new data
	xxxxxxxx1xxxxxxx	Client 8 is active with new data
	xxxxxxx1xxxxxxxx	Client 9 is active with new data
	xxxxxx1xxxxxxxxx	Client 10 is active with new data
	xxxxx1xxxxxxxxxx	Client 11 is active with new data
	xxxx1xxxxxxxxxxx	Client 12 is active with new data
	xxx1xxxxxxxxxxxx	Client 13 is active with new data
	xx1xxxxxxxxxxxxx	Client 14 is active with new data
	x1xxxxxxxxxxxxxx	Client 15 is active with new data
	1xxxxxxxxxxxxxxx	Client 16 is active with new data
	All other options	Reserved
15–0	Receive Queue Disable.	
RQD	xxxxxxxxxxxxxxx1	Disable receive queue for Client 1
	xxxxxxxxxxxxxx1x	Disable receive queue for Client 2

Table continues on the next page...

Table 78-11. NAR_CDR field descriptions (continued)

Field	Description	
	xxxxxxxxxxxxx1xx	Disable receive queue for Client 3
	xxxxxxxxxxxx1xxx	Disable receive queue for Client 4
	xxxxxxxxxxx1xxxx	Disable receive queue for Client 5
	xxxxxxxxxx1xxxxx	Disable receive queue for Client 6
	xxxxxxxxx1xxxxxx	Disable receive queue for Client 7
	xxxxxxxx1xxxxxxx	Disable receive queue for Client 8
	xxxxxxx1xxxxxxxx	Disable receive queue for Client 9
	xxxxxx1xxxxxxxxx	Disable receive queue for Client 10
	xxxxx1xxxxxxxxxx	Disable receive queue for Client 11
	xxxx1xxxxxxxxxxx	Disable receive queue for Client 12
	xxx1xxxxxxxxxxxx	Disable receive queue for Client 13
	xx1xxxxxxxxxxxxx	Disable receive queue for Client 14
	x1xxxxxxxxxxxxxx	Disable receive queue for Client 15
	1xxxxxxxxxxxxxxx	Disable receive queue for Client 16
	All other options	Reserved

78.4.8 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following table shows the format of the NAR_AHFPAR.

Register index: E

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0			0		

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	AHB0 1kEN	0	0	0	0	0	0	AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_AHFPAR is described in the following table.

Table 78-12. NAR_AHFPAR field description

FIELD	Description
31-11	Reserved and must be written with 0s
10 AHB01kEN	Enable 1K Marker Event
9-4	Reserved and must be written with 0s
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Overlay and internal trace memory full status generation **Note:** memory ranges from 1K to 15K

78.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

78.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/ filtering when two output ports are enabled, enabling timestamp and so on.

4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note

The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see Suppress mode. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

78.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in the following table.

Table 78-13. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

78.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

78.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

78.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

78.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

78.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in the following table on the incoming messages before transmitting to one of the output ports.

Table 78-14. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8 × 32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8 × 32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

78.5.3 Data reception and arbitration

The NAR supports 7 clients. Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There are two types of clients supported at NAR client interface. The client types have their own protocols to send Nexus messages. The only common parameter is the Nexus word size (beat). One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits. Internally, the NAR converts all client interfaces to a common interface. To achieve this, the NAR instantiates interface level gaskets that handle the protocol conversion and clock domain transfer.

When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn × 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

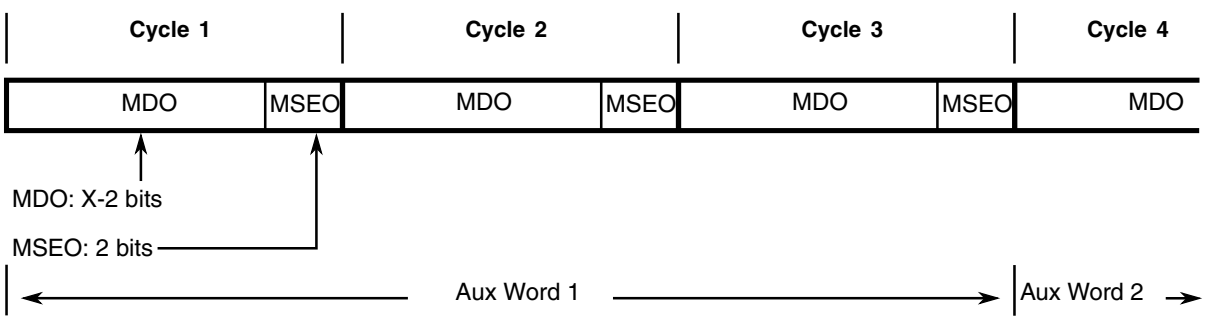


Figure 78-2. AUX bus parsing

Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give

preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

78.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

78.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

78.5.6 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

78.5.6.1 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

78.5.7 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)

- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 × 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

78.5.8 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- Watchpoint—The NAR generates its own watchpoint match message under the circumstances shown in Table 78-18.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 78-15. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value

Table continues on the next page...

Table 78-15. NAR message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message[1]	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC [2]	NAR source ID
	4	4	fixed	ETYPE	See Table 78-16
	12	12	fixed	ECODE	See Table 78-17
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message[3]	6	6	fixed	TCODE	TCODE value = 15
	6	6	fixed	SRC [2]	NAR source ID
	6	6	fixed	WPHIT	See Table 78-18
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2012 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2012 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

78.5.8.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in the following table.

Table 78-16. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see Table 78-17.
0011 (h'3)	Trace memory bus port read/ write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.[2]
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See Table 78-17	The specific configuration error is conveyed in bits 0–3 of the ECODE field (Table 78-17), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1101 (h'D)	Receive Queue Overrun	Minor	Drop message	If a client sends a data word that the NPC cannot accept (due to a full receive queue), the new data word will be dropped2.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

2. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in the following table.

Table 78-17. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.

Table continues on the next page...

Table 78-17. Configuration error codes (continued)

ECODE	ETYPE	Description	NAR action	Comment
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

78.5.8.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

Table 78-18. Watchpoint hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

78.5.9 Timestamp function

The NAR and its clients may transmit timestamps to allow precise ordering of events. Synchronous or asynchronous clients use the NAR timer value broadcasted over a bus to ensure that all timestamps are in sync as shown in the following figure. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

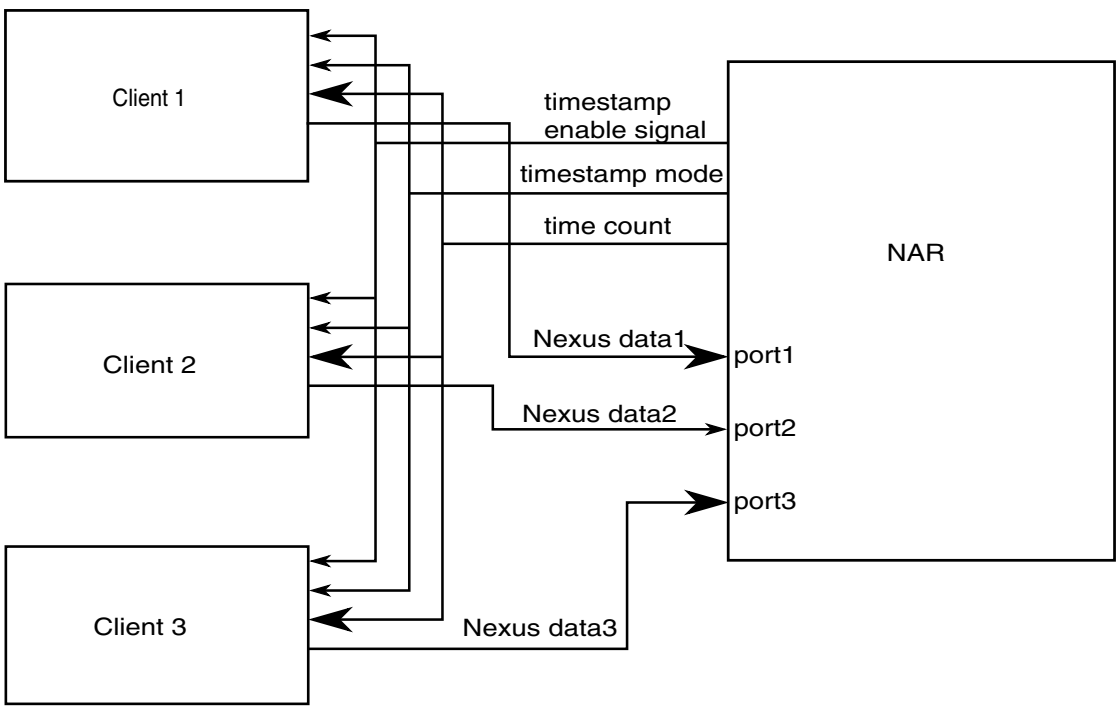


Figure 78-3. Timestamp architecture

Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The timestamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode, and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

78.5.10 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:

- AHB start (base) address—Base address = 0x0D00_0000 (i.e.TBAHI = 0x0, TBALO = 0x0D00_0000
- Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)

- Thus, the total block size is (0xC0 × 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

78.5.11 Configuration/debug interface JTAG

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in Table 78-19. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

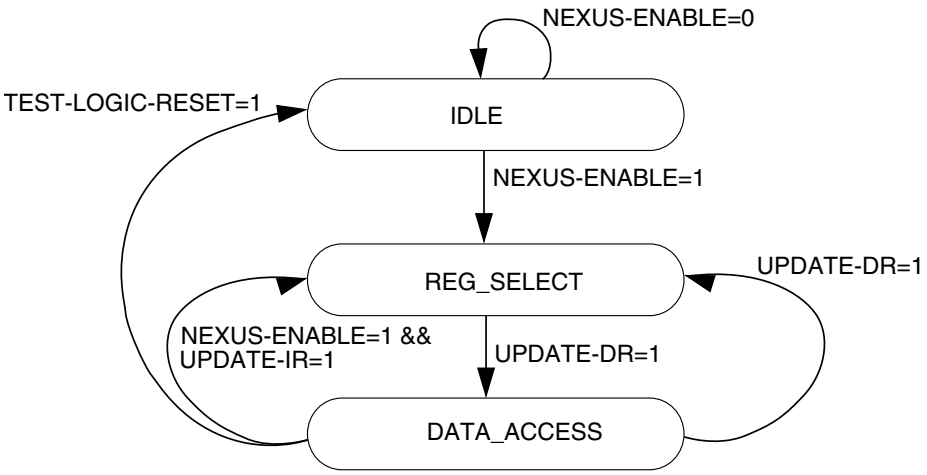


Figure 78-4. Nexus controller state machine

Table 78-19. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

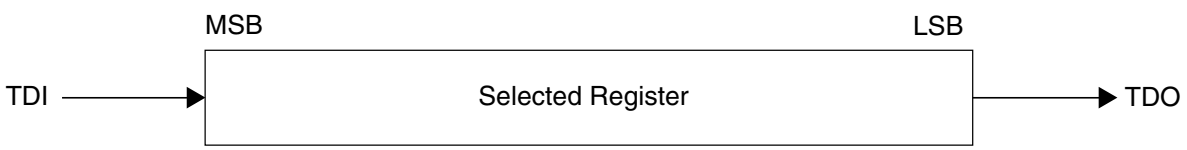


Figure 78-5. Shifting data into register

78.5.11.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

78.5.11.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 78-4, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 78-20. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

78.5.11.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following figure. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in Table 78-1.

Table 78-21. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state.When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state.When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 78-22. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

78.5.12 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface

- Early Write (for higher BW in special module)
- 32-bit address bus width
- 64-bit data bus width
- Transfer Types
- IDLE, NONSEQ, SEQ
- Single Transfers
- Burst Transfers
- Configurable support for INCR
- Configurable support for INCR4, INCR8, INCR16, WRAP4
- Response Type
- OKAY, ERROR
- Transfer Size
- Byte, Half Word, Word, Double-Word
- Protection Control
- Cacheable/ Non Cacheable Access Control
- User/Privilege Access

- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

78.5.12.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

78.5.12.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

78.5.12.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

78.5.13 Legacy client interface controller

This block converts the legacy client protocol into internal client protocol that is accepted by the internal NPC block. The Nexus beat size must be same for both interfaces. The Nexus beat is composed of MDO bits along with two MSEO bits. The trace data width from the legacy client is fixed at one beat. The architecture of this block is shown in Figure 78-6.

The legacy client sends each message after arbitrating the bus. It requests the gasket with a 2-bit request signal whenever it has data to send. A 00 means no request, 01 a low-priority request, 10 a medium-priority request, and 11 a high-priority request. The legacy client can send one maximum message sized data on receiving a grant from the block. There is no backpressure mechanism during transfer of a message from the client to this block. The block asserts a grant if it has at least one max message size space in its FIFO. Seeing the grant the client asserts busy signal and holds it until it wants to send data, it may hold it up to max message size data is transferred.

The following architecture describes the legacy client to internal client protocol conversion block. There is a small FIFO of paramaterizeable depth used to store the data from client. The block asserts a grant only when at least one max message sized space is available in FIFO. The write happens in the client's clock domain. The write pointer is passed to other domain which is NPC clock domain.

The logic on other side compares read and write pointer to check the data availability in the block's FIFO and space availability in NPC's receive queue. If space and data both are available it tries to send message to the NPC.

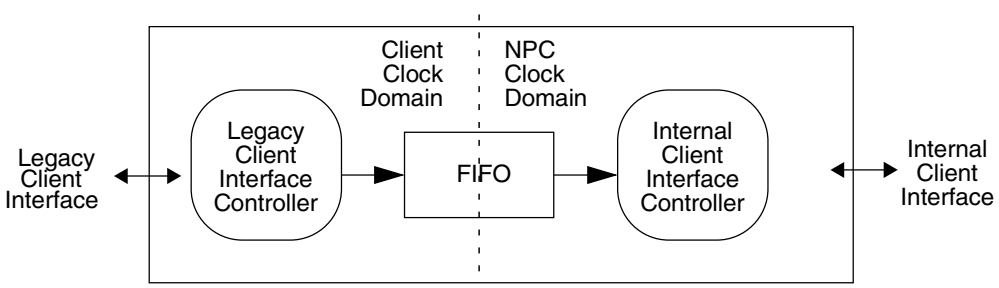


Figure 78-6. Legacy client Interface controller

The size of the receive queue of the corresponding client at NPC must be known to the block. The block drives a not empty signal whenever it has data in FIFO and looks for a ready signal. It can send as many Nexus beats as receive queue can store. It operates on decrements a counter after sending data to the NPC so that it can track how much data it can send. The counter is incremented by the value of acknowledge from NPC which tells how may spaces are freed in receive queue.

78.6 Application information

The base application for the NAR is collecting and collating Nexus trace data.

Example flow of NAR configuration for different cases.

Configuration with Two output port enabled

1. For Two port enabled one port must be Message Select (MSP) and other Alternate (ALT); Trace Memory and Aurora ports with multiple clients

   Selection of MSP port depends on user choice. One option could be based on the requirement of Filtering. If Messages from some known sources with some specific type (if any) is required to be sent to a particular port then select user can select that port as MSP. All other messages will go to the Alternate port. The filter can be used as inverted manner where unmatched messages will go to MSP. (Please see section "Output arbitration" for details)
   - Configure the Source Filter register (NAR_SFR), To enable proper filtering of messages based on Message source going to MSP or ALT port
     - Example: for DWPU connected in Nexus out port set the source ID of the core and NXMC in any two source filter type and enable it
   - Configure the Type Filter register (NAR_TFR), To enable proper filtering of messages based on Message Type going to MSP or ALT port
     - Example: for DWPU connected in Nexus out port set the Type of messages in any two Type filter type and enable it
   - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
     - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
     - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
   - Configure Client disable register NAR_CDR (disable clients which are not required)
   - Configure NPC suppress trigger registers NAR_STCR[x] (x=0-3) (suppress the clients for which data is not to be forwarded at output) (Difference between suppressing and disabling is that suppressing can be done dynamically during NAR operation but disabling can't be done this way which may leads to system hang)
   - Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
   - Configure the NAR control register for other controls and Enable NAR
     - Set the MSP as Nexus Out and ALT as Trace memory
     - Set AQP for partitioning the Main Queue for ALT and MSP ports

- Set time stamp controls
- Enable NAR to start operation
- Configuration for Single Output port with single partition in NAR central Message Queue

2. Only Trace Memory port is enabled with multiple client
   - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
     - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
     - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
   - Configure Client disable register NAR_CDR (disable clients which are not required)
   - Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
   - Configure the NAR control register for other controls and Enable NAR
     - Set the Trace Memory port as ALT port (Alternate port is only the main port in single out port application)
     - Set AQP for disabling partitioning
     - Set time stamp controls
     - Set watch point controls
     - Enable NAR to start operation

Chapter 79
Nexus Module

79.1 Introduction

The e200z425Bn3Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

79.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

79.1.2 Terms and Definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 79-1. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

79.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Watchpoint Event output pin (**nex_evto_b**)
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

Note

For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations

(not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

79.1.4 Functional Block Diagram

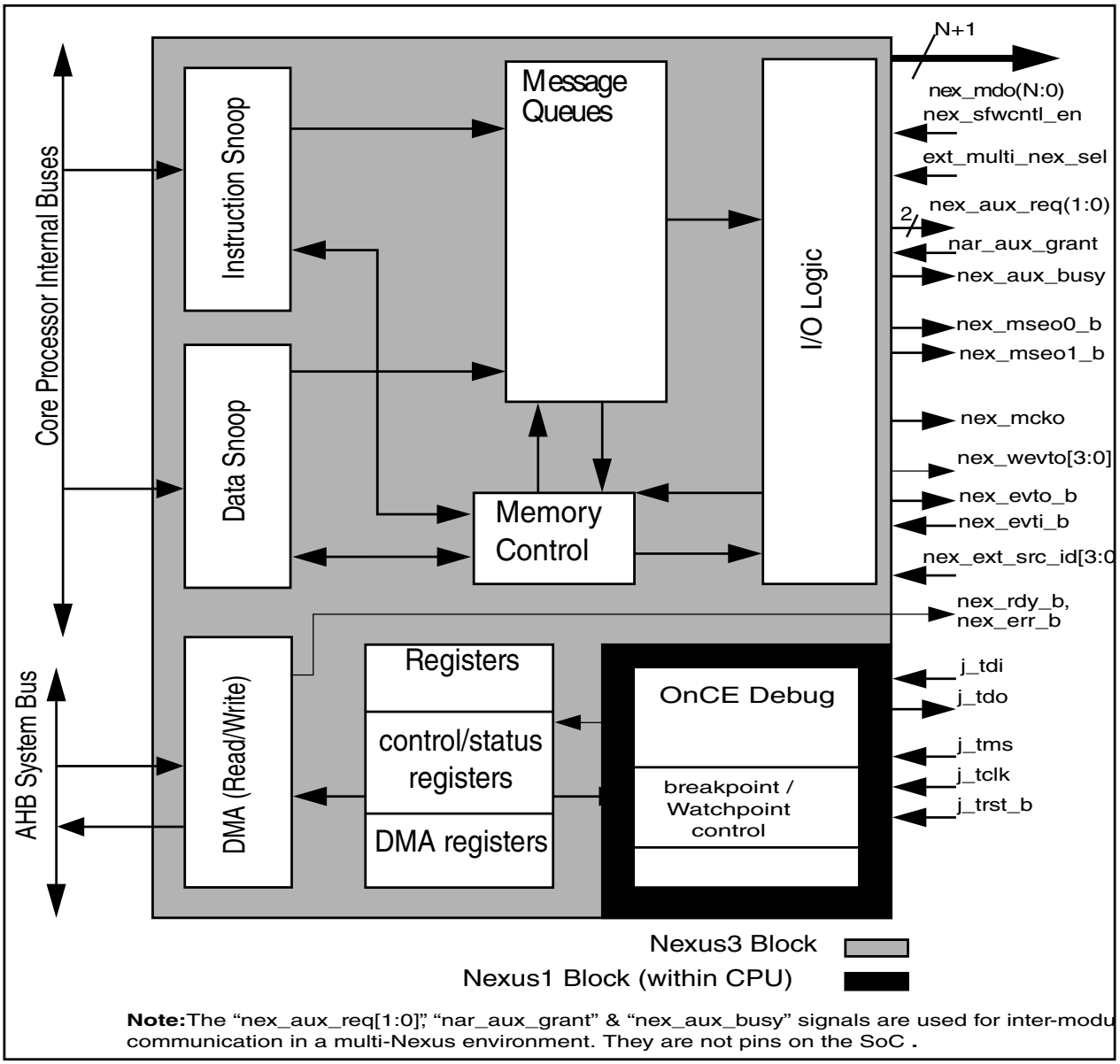


Figure 79-1. Nexus 3 functional block diagram

79.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

79.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

79.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001 2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 79-2.

Table 79-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 79-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 79-7)

Table continues on the next page...

Table 79-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 79-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace -	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier

Table continues on the next page...

Table 79-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Read Message w/ Sync	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 79-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 79-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 79-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)

Table continues on the next page...

Table 79-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 79-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 79-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 79-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 79-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost

Table continues on the next page...

Table 79-4. Error Type (ETYPE) Encoding (TCODE = 8) (continued)

Error Type	Description
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 79-5 shows the encodings used for resource codes for certain messages.

Table 79-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 79-6 shows the event code encodings used for certain messages.

Table 79-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 79-7 shows the data trace size encodings used for certain messages.

Table 79-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)

Table continues on the next page...

Table 79-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14) (continued)

DTM Size Encoding	Transfer Size
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

79.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 79-8. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	

Table continues on the next page...

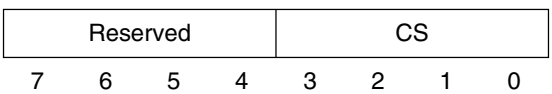
Table 79-8. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

79.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# - 0x1; Read-only; Reset - 0x0

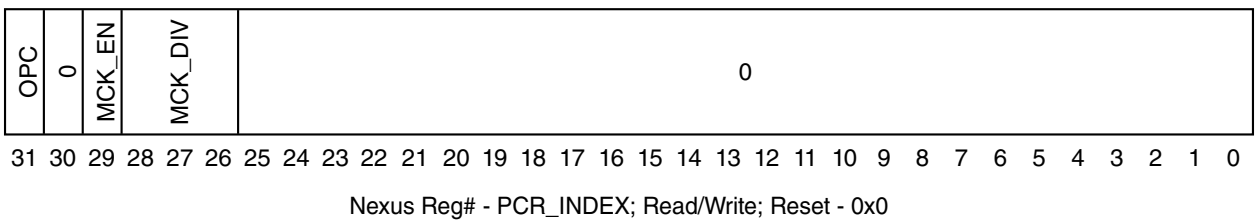
Figure 79-2. Client Select Control (CSC) register

Table 79-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

79.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.



Nexus Reg# - PCR_INDEX; Read/Write; Reset - 0x0

Figure 79-3. Port Configuration Register

Table 79-10. PCR field descriptions

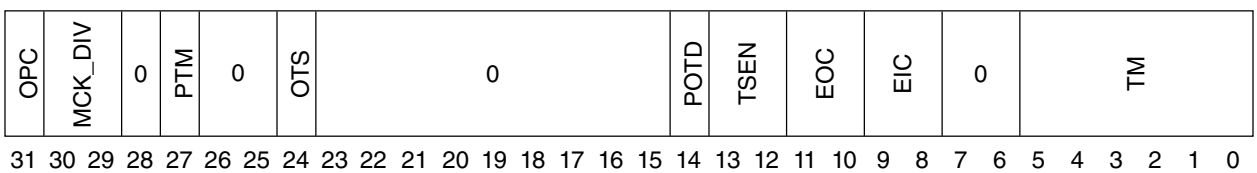
Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

79.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 79-4 and its fields are described in Table 79-11.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 79-4. Development Control Register 1

Table 79-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality

Table continues on the next page...

Table 79-11. DC1 field descriptions (continued)

Bits	Name	Description
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **nex_evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1] 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.

Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

79.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z425Bn3) Core Debug Support chapter.

Development Control Register 2 is shown in Figure 79-5 and its fields are described in Table 79-12.

Figure 79-5. Development Control Register 2

Table 79-12. DC2 field descriptions

Bits	Name	Description
15:0	EWC	EVTO Watchpoint Configuration[1] 0000000000000000 No Watchpoints #0-15 trigger **nex_evto_b** XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **nex_evto_b** XXXXXXXXXXXXXX1X Watchpoint #1 triggers **nex_evto_b** XXXXXXXXXXXXX1XX Watchpoint #2 triggers **nex_evto_b** XXXXXXXXXXXX1XXX Watchpoint #3 triggers **nex_evto_b** XXXXXXXXXXX1XXXX Watchpoint #4 triggers **nex_evto_b** XXXXXXXXXX1XXXXX Watchpoint #5 triggers **nex_evto_b** XXXXXXXXX1XXXXXX Watchpoint #6 triggers **nex_evto_b** XXXXXXXX1XXXXXXX Watchpoint #7 triggers **nex_evto_b** XXXXXXX1XXXXXXXX Watchpoint #8 triggers **nex_evto_b** XXXXXX1XXXXXXXXX Watchpoint #9 triggers **nex_evto_b** XXXXX1XXXXXXXXXX Watchpoint #10 triggers **nex_evto_b** XXXX1XXXXXXXXXXX Watchpoint #11 triggers **nex_evto_b** XXX1XXXXXXXXXXXX Watchpoint #12 triggers **nex_evto_b** XX1XXXXXXXXXXXXX Watchpoint #13 triggers **nex_evto_b** X1XXXXXXXXXXXXXX Watchpoint #14 triggers **nex_evto_b** 1XXXXXXXXXXXXXXX Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 79-6 and its fields are described in Table 79-13.

Figure 79-6. Development Control 3 (DCR3) register

Table 79-13. DC3 field descriptions

Bits	Name	Description
15:14	—	Reserved for watchpoint expansion

Table continues on the next page...

Table 79-13. DC3 field descriptions (continued)

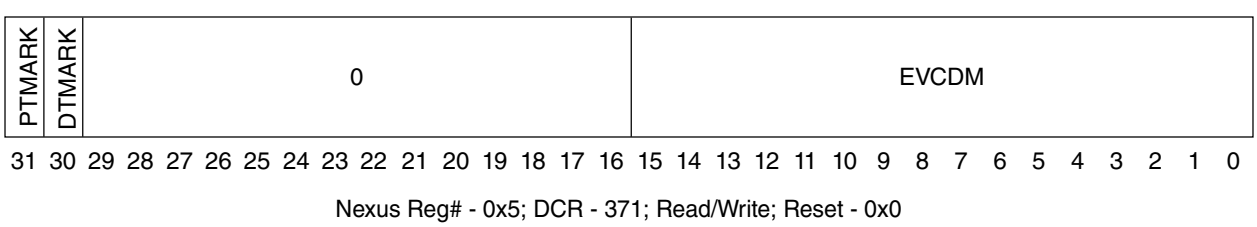
Bits	Name	Description
13:0	EWC	EVTO Watchpoint Configuration[1]
		00000000000000 No Watchpoints #16-#29 trigger **nex_evto_b**
		XXXXXXXXXXXXX1 Watchpoint #16 triggers **nex_evto_b**
		XXXXXXXXXXXX1X Watchpoint #17 triggers **nex_evto_b**
		XXXXXXXXXXX1XX Watchpoint #18 triggers **nex_evto_b**
		XXXXXXXXXX1XXX Watchpoint #19 triggers **nex_evto_b**
		XXXXXXXXX1XXXX Watchpoint #20 triggers **nex_evto_b**
		XXXXXXXX1XXXXX Watchpoint #21 triggers **nex_evto_b**
		XXXXXXX1XXXXXX Watchpoint #22 triggers **nex_evto_b**
		XXXXXX1XXXXXXX Watchpoint #23 triggers **nex_evto_b**
		XXXXX1XXXXXXXX Watchpoint #24 triggers **nex_evto_b**
		XXXX1XXXXXXXXX Watchpoint #25 triggers **nex_evto_b**
		XXX1XXXXXXXXXX Watchpoint #26 triggers **nex_evto_b**
		XX1XXXXXXXXXXX Watchpoint #27 triggers **nex_evto_b**
		X1XXXXXXXXXXXX Watchpoint #28 triggers **nex_evto_b**
		1XXXXXXXXXXXXX Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

79.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 79-7 and its fields are described in Table 79-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

Figure 79-7. Development Control 4 (DC4) register

Table 79-14. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark

Table continues on the next page...

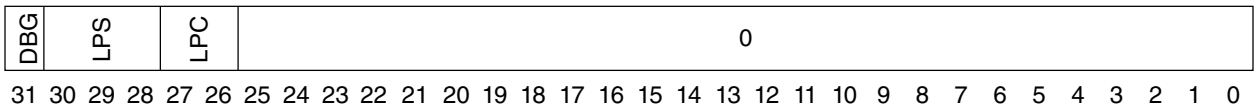
Table 79-14. DC4 field descriptions (continued)

Bits	Name	Description
		0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask[1] 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages 1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

1. Refer to Table 5 for implemented EVCODEs.
1. Refer to Table 5 for implemented EVCODEs.

79.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 79-8. Development Status Register

Table 79-15. DS field description

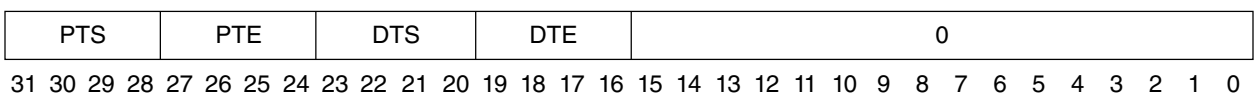
Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

79.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that

direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z425Bn3) Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 79-9. Watchpoint Trigger (WT) Register

Table 79-16 details the Watchpoint Trigger register fields.

Table 79-16. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	PTE - Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . .

Table continues on the next page...

Table 79-16. WT field descriptions (continued)

Bits	Name	Description
		1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 79-10. Program Trace Start Trigger Control (PTSTC) register

Table 79-17 details the PTSTC register fields.

Table 79-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4

Table 79-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 79-11. Program Trace End Trigger Control (PTETC) register

Table 79-18 details the PTETC register fields.

Table 79-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

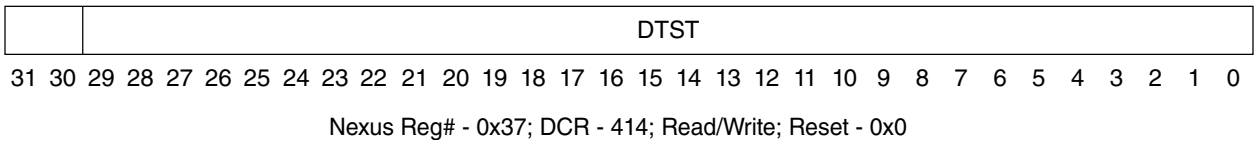


Figure 79-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 79-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27

Table 79-19. DTSTC field descriptions

Bits	Name	Description
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 038; DCR - 415; Read/Write; Reset - 0x0

Figure 79-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 79-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19

Table 79-20. DTET field descriptions

Bits	Name	Description
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

79.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 79-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 79-21. WMSK field descriptions

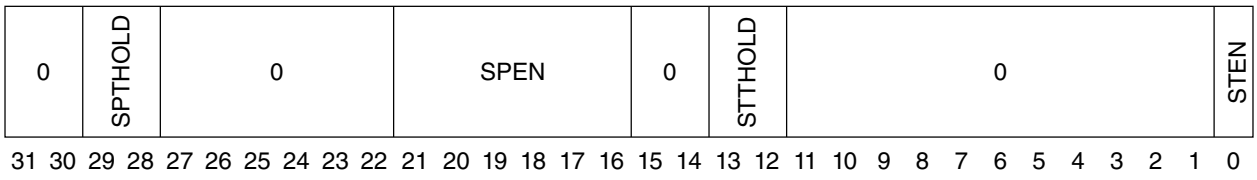
Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging
		00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM

Table 79-21. WMSK field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

79.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.



Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 79-15. Nexus Overrun Control Register

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 79-16. Nexus Overrun Control Register Fields

79.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

RWT1	RWT2	RWT3	RWT4	0	STDC1	STDC2	STDC3	STDC4	0	RC1	RC2	RC3	RC4	DI	0
31 30	29 28	27 26	25 24	23 22 21 20 19 18 17 16	15	14	13	12	11 10 9 8	7	6	5	4	3	2 1 0

Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 79-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 79-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4

Table continues on the next page...

Table 79-22. DTC field descriptions (continued)

Bits	Name	Description
		0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core (e200z425Bn3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

79.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 79-18. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 79-19. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 79-20. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 79-21. Data Trace Start Address 4 (DTSA4) register

79.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 79-22. Data Trace End Address 1 (DTEA1) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 79-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 79-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 79-25. Data Trace End Address 4 (DTEA4) register

Table 79-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 79-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

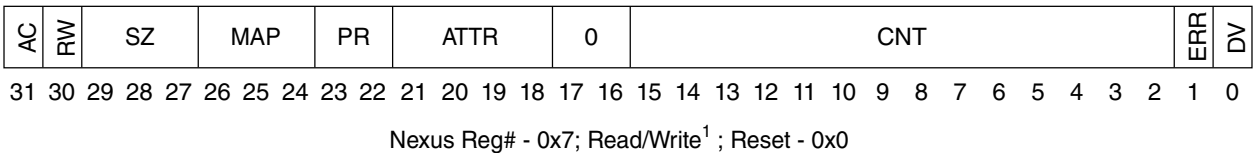
Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

79.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 79-25.



Nexus Reg# - 0x7; Read/Write[1] ; Reset - 0x0

Figure 79-26. Read/Write Access Control/Status (RWCS) register

NOTES:

[1]ERR and DV are read-only

Table 79-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality

Table continues on the next page...

Table 79-24. RWCS field description (continued)

Bits	Name	Description
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR	Read/Write Access Error (see Table 79-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 79-25)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 79-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

79.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

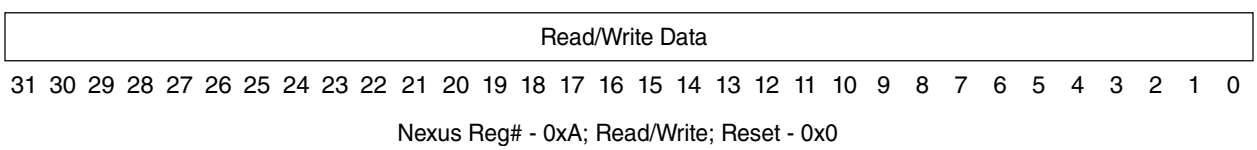


Figure 79-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 79-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 79-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 79-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 79-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

79.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Read/Write Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x9; Read/Write; Reset - 0x0

Figure 79-28. Read/Write Access Address (RWA) register

79.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 79-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 79-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

(7bits)	(1 bit)
Nexus Register Index	R/W

RESET Value: 0x0

Nexus Register Index	Selected from values in Table 79-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

79.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 79-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

79.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

79.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

79.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

79.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

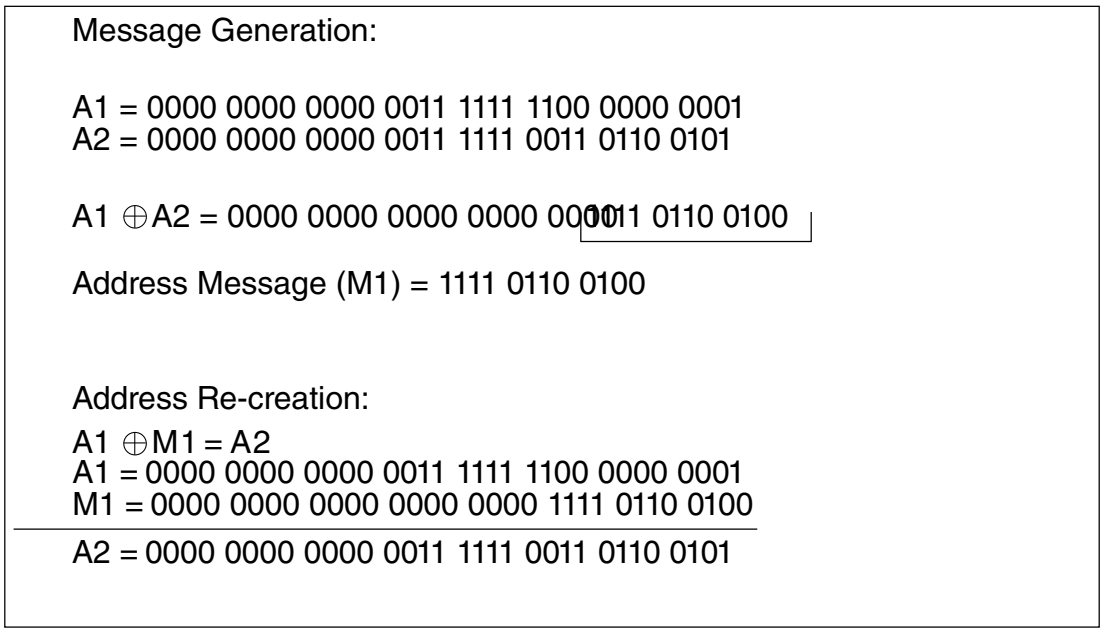


Figure 79-29. Relative address generation and recreation

79.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

79.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

79.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

79.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

79.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in

Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

79.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 79-28 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 79-28.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 79-28. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message

Table continues on the next page...

Table 79-28. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

79.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

79.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

79.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

79.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

79.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

79.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

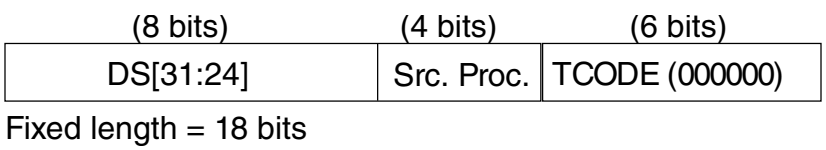


Figure 79-30. Debug Status message format

79.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 79-3 and Table 79-4):

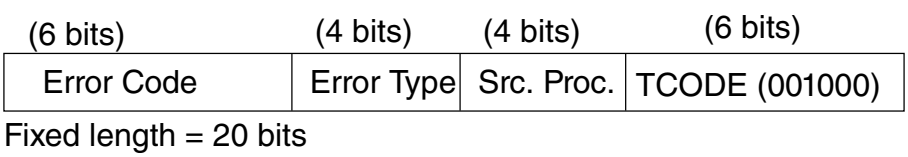


Figure 79-31. Error message format

79.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

79.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

79.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z425Bn3) processors contain a *PowerISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

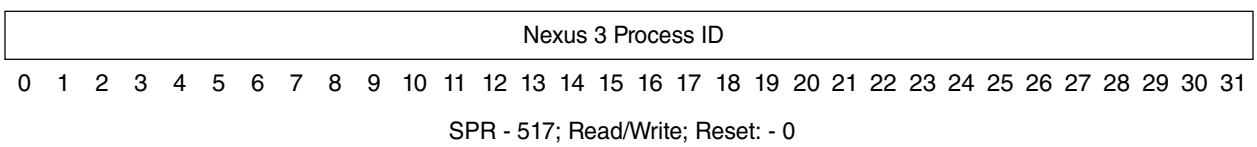


Figure 79-32. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for

PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.

- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

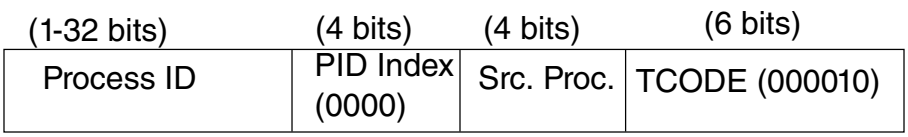


Figure 79-33. Ownership Trace Message format

79.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the e200z425Bn3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z425Bn3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

79.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

79.12.1.1 Indirect Branch Message instructions

Table 79-29 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 79-29. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

79.12.1.2 Direct Branch Message Instructions

Table 79-30 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 79-30. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

79.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

79.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

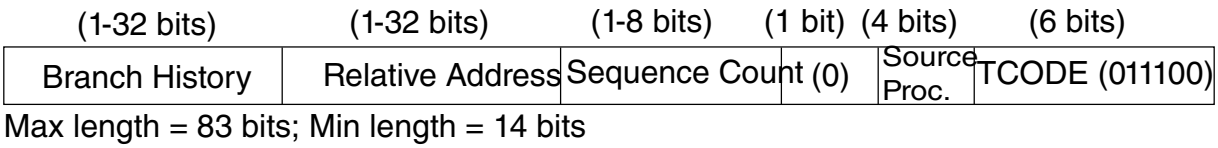
Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

79.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

79.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

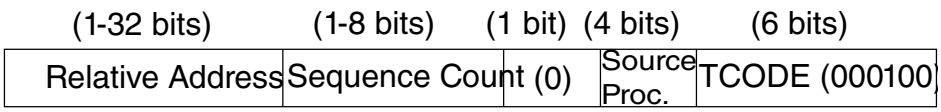


Max length = 83 bits; Min length = 14 bits

Figure 79-34. Indirect branch message (history) format

79.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:



Max length = 51 bits; Min length = 13 bits

Figure 79-35. Indirect branch message format

79.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

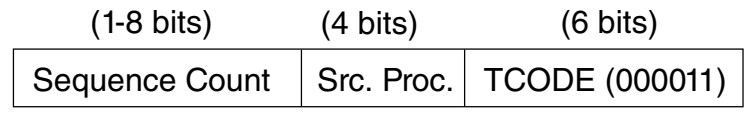


Figure 79-36. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

79.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

79.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

79.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 79-31. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

79.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

79.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

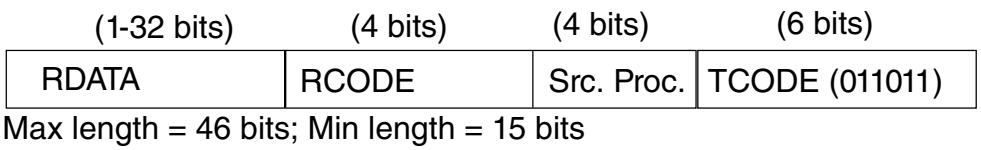


Figure 79-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 79-32. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

79.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 79-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

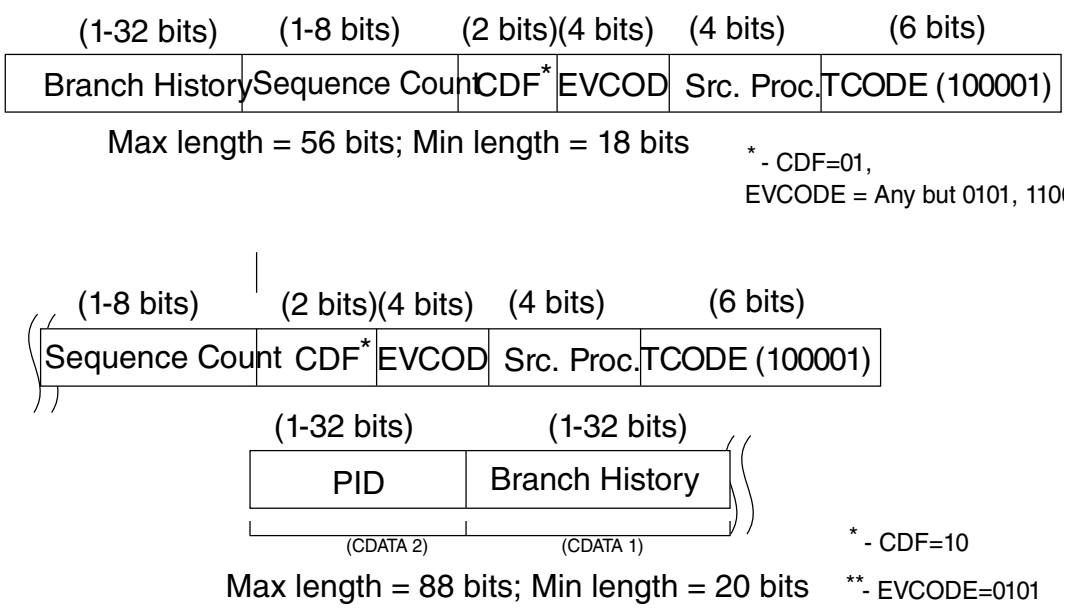


Figure 79-38. Program Correlation message formats

If timestamping is enalbed, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

79.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

79.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

79.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 79-33):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.

- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Program Trace Synchronization are summarized in Table 79-33.

Table 79-33. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

79.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 79-33):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

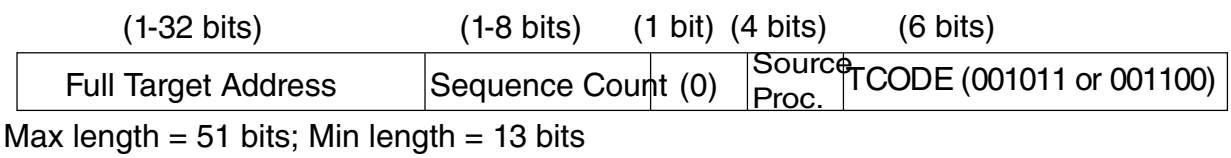


Figure 79-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

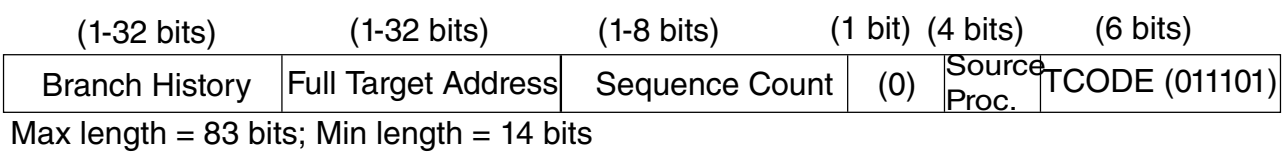


Figure 79-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 79-34.

Table 79-34. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

79.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

79.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

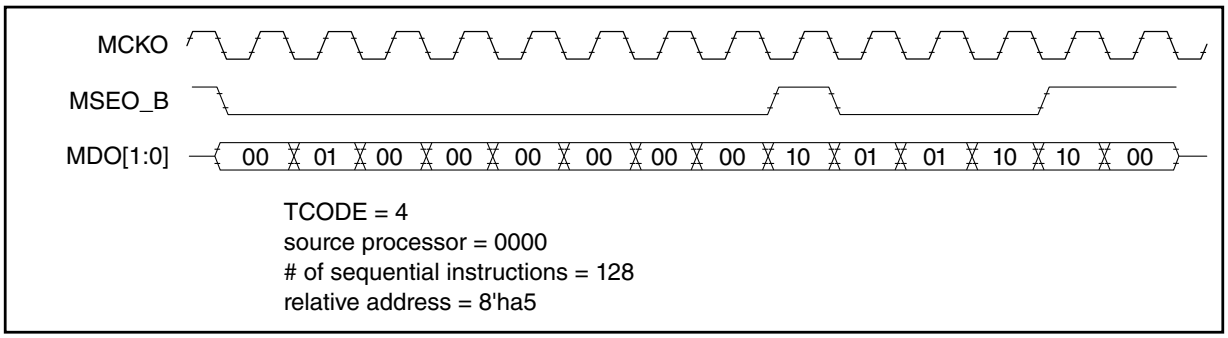


Figure 79-41. Program Trace - Indirect Branch message (Traditional)

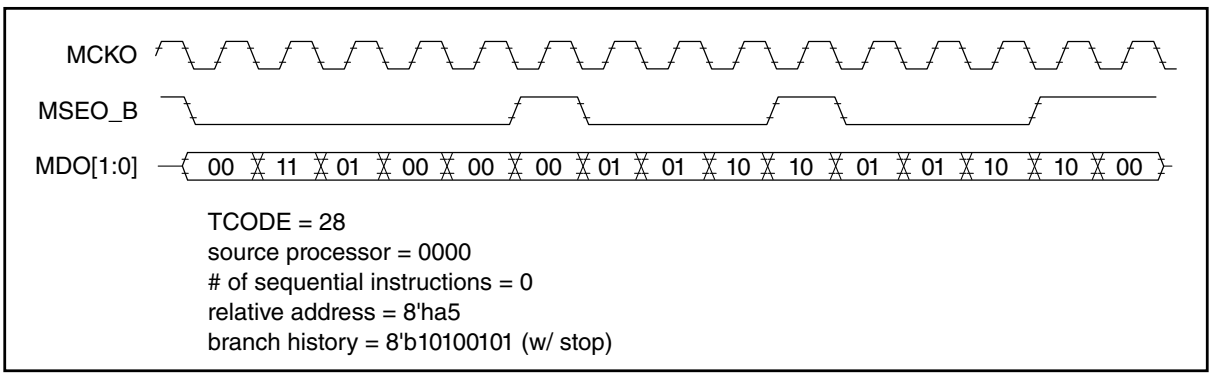


Figure 79-42. Program Trace - Indirect Branch message (history)

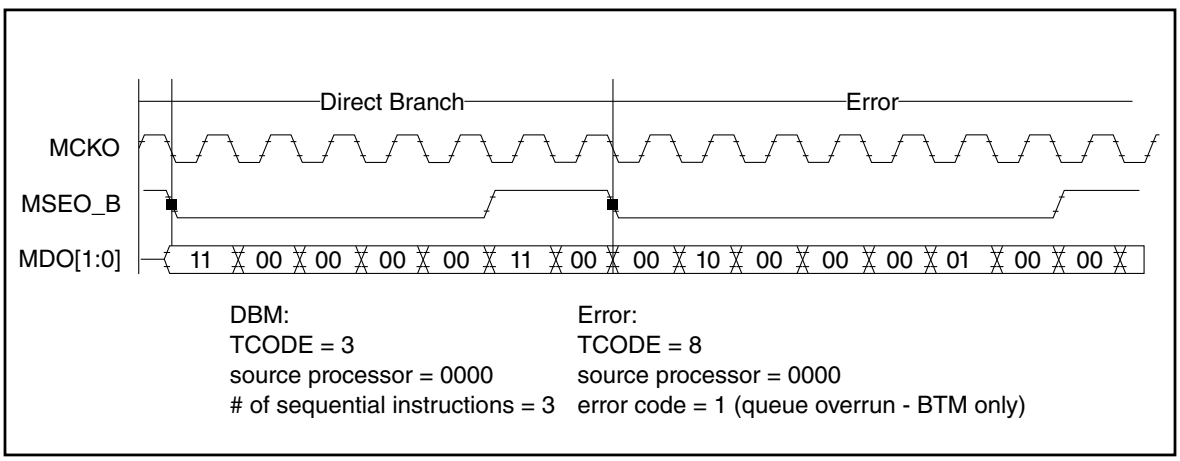


Figure 79-43. Program Trace - Direct Branch (traditional) & Error messages

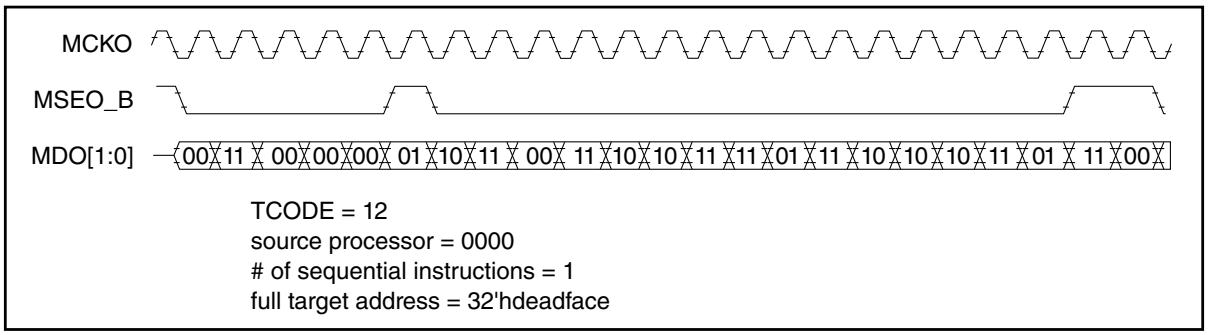


Figure 79-44. Program Trace - Indirect Branch w/ Sync. message

79.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

79.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z425Bn3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).

79.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

79.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 79-45. Data Write message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

79.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 79-46. Data Read fessage format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Note

* Core (e200z425Bn3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** =

0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z425Bn3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

79.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 79-35):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

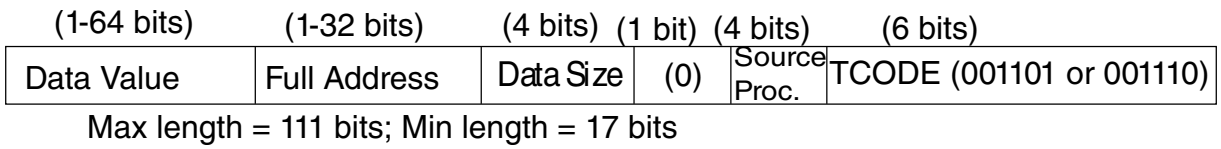


Figure 79-47. Data Write/Read with Sync. message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Data Trace Synchronization are summarized in Table 79-35.

Table 79-35. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

79.13.3 DTM Operation

79.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

79.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

79.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

79.13.3.4 Bus Cycle special cases

Table 79-36. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)

Table continues on the next page...

Table 79-36. Bus Cycle Special Cases (continued)

Special Case	Action
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z425Bn3) Core Debug Support chapter.)

Note

A store to the store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

79.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

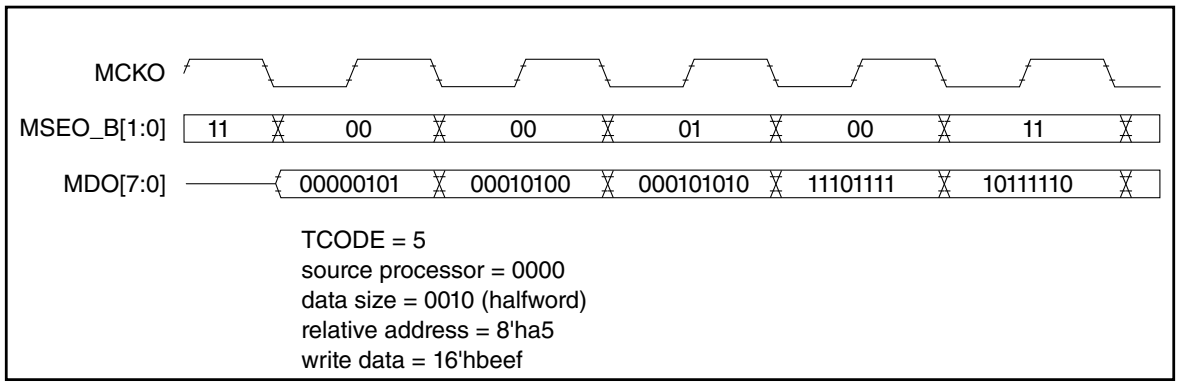


Figure 79-48. Data Trace - Data Write Message

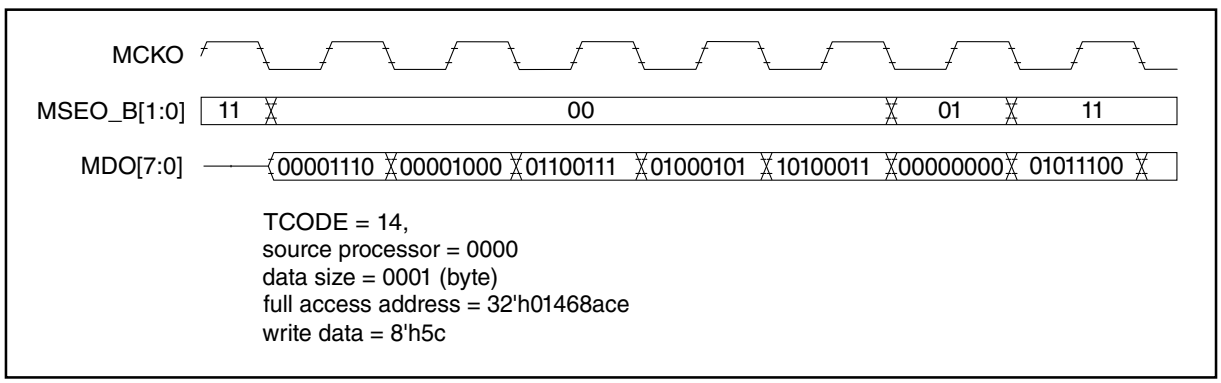


Figure 79-49. Data Trace — Data Read w/ Sync message

79.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

79.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

79.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

79.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core (e200z425Bn3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

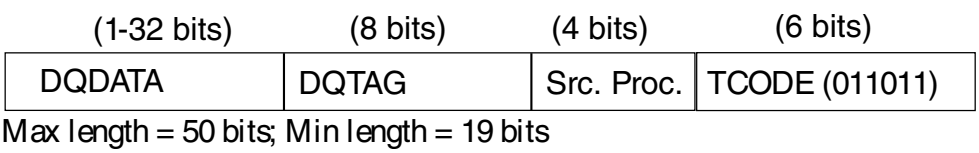


Figure 79-50. Data Acquisition message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

79.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z425Bn3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 79-39 for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

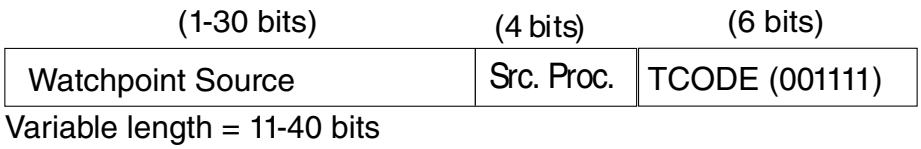


Figure 79-51. Watchpoint message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 79-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
000000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX - Watchpoint #3 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX - Watchpoint #4 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX - Watchpoint #5 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX - Watchpoint #6 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX - Watchpoint #7 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX - Watchpoint #8 enabled for WTM	
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX - Watchpoint #9 enabled for WTM	
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX - Watchpoint #10 enabled for WTM	
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX - Watchpoint #11 enabled for WTM	
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX - Watchpoint #12 enabled for WTM	
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX - Watchpoint #13 enabled for WTM	

Table 79-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX	Watchpoint #14 enabled for WTM
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX	Watchpoint #15 enabled for WTM
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX	Watchpoint #16 enabled for WTM
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX	Watchpoint #17 enabled for WTM
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX	Watchpoint #18 enabled for WTM
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX	Watchpoint #19 enabled for WTM
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX	Watchpoint #20 enabled for WTM
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX	Watchpoint #21 enabled for WTM
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #22 enabled for WTM
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #23 enabled for WTM
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #24 enabled for WTM
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #25 enabled for WTM
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #26 enabled for WTM
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #27 enabled for WTM
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #28 enabled for WTM
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #29 enabled for WTM

79.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

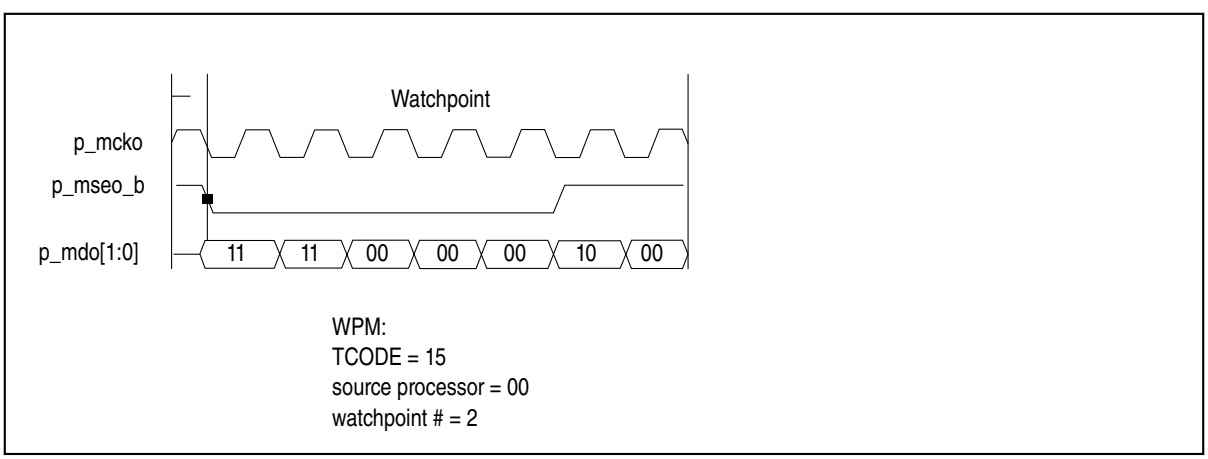


Figure 79-52. Watchpoint Message & Watchpoint Error message

79.16 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

79.16.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:

- Access Control (AC) → 1b'1 (to indicate start access)
- Map Select (MAP) → 3b'000 (primary memory map)
- Access Priority (PR) → 2b'11 (highest priority)
- Read/Write (RW) → 1b'1 (write access)
- Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
- Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data (RWD) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD register to the memory mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 79-39 for details) and clears the ERR bit in the RWCS register. Otherwise, if the access completes with an error, the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once the DV bit has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start**, and **nex_rdy_b** pins, as well as the AC, DV, and ERR bits within the RWCS, provide Read/Write Access status to the external development tool.

79.16.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block then arbitrates for the AHB system bus and transfers the first data value from the RWD register to the memory-mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output is asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA register is incremented to the next word size (specified in the SZ field) the number from the CNT field is decremented. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS register. Otherwise, the ERR bit is set and the DV bit is cleared to indicate an error has occurred, , the block transfer is aborted, and the AC bit in the RWCS register is cleared. Clearing the DV bit indicates that the device is ready for the next access in the block transfer, or that an error has occurred.
3. If the AC bit has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS register are not changed when executing a block write access. The original values can be read by the external development tool at any time.

79.16.3 Single read access

1. Initialize the Read/Write Access Address (RWA)register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)

- Access Priority (PR) → 2b'11 (highest priority)
- Read/Write (RW) → 1b'0 (read access)
- Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
- Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

An Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 79-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred. The **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit is cleared in either case.
4. The data can then be read from the Read/Write Access Data register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start**and **nex_rdy_b** pins as well as the AC, DV, and ERR bits within the RWCS register provide Read/Write Access status to the external development tool.

79.16.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the

RWCS register and asserts the **nex_rdy_b** pin (see Table 79-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS register, and asserts the **nex_rdy_b** pin (see Table 79-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the RWD register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

79.16.5 Error handling

The Nexus 3 module handles various error conditions as follows:

79.16.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write error

79.16.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
    a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
    b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

79.16.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

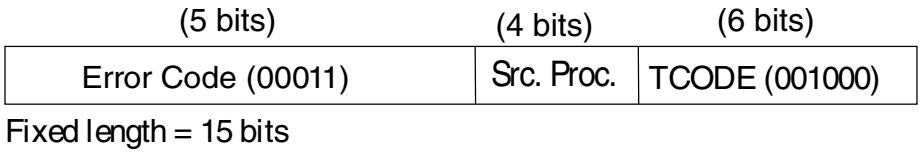


Figure 79-53. Error message format1

79.17 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

79.17.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_evto_b** pin, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 79-39. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core (e200z425Bn3) Core Debug Support chapter.

Table 79-38. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 79-39.

Table 79-39. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.
nex_ahb_start	O	AHB Start (**nex_ahb_start**) is an output pin used to indicate to the external tool or SoC that the Nexus block is requesting the next Read/Write Access on the system bus. If Nexus is enabled, this signal is asserted upon an acknowledged request to start an AHB system bus transfer (Nexus read or write) and is pulsed asserted for one **nex_clk** clock period, corresponding to the first clock of the address phase of the transfer. Upon exit from system reset or if Nexus is disabled, **nex_ahb_start** remains de-asserted.
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted.
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 79-40. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 79-42.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

79.17.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 79-41. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the .

Table 79-41. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 79-54 illustrates the state diagram for single pin MSEO transfers.

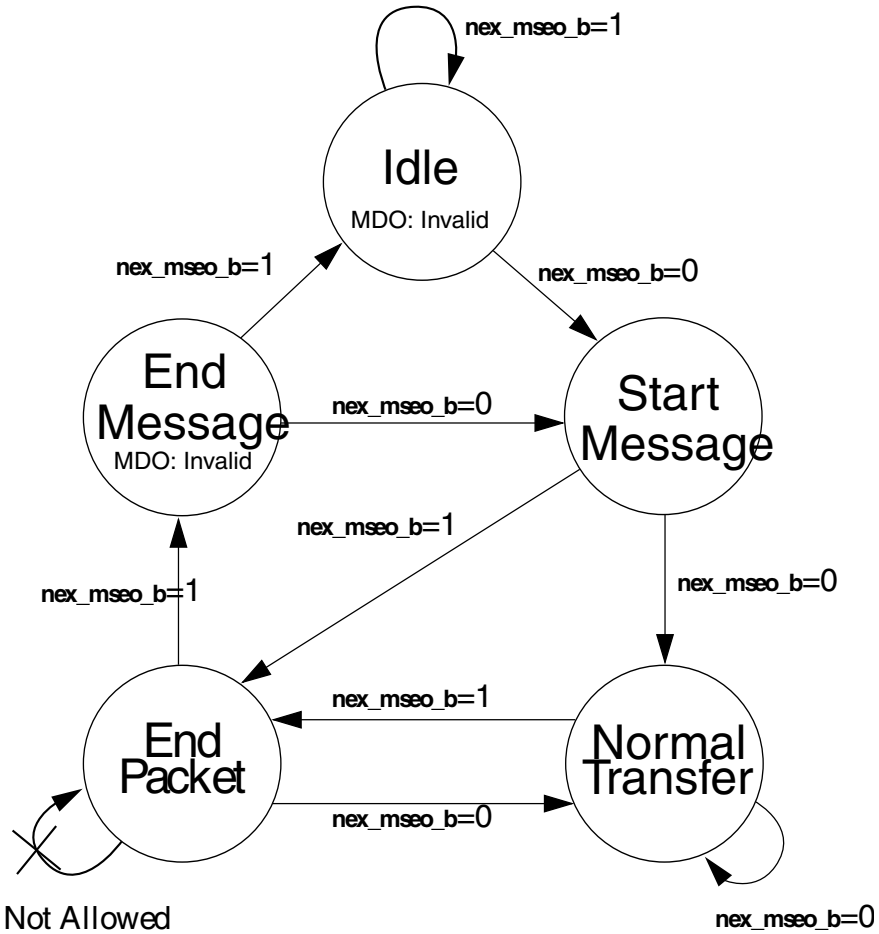


Figure 79-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 79-55 illustrates the state diagram for dual pin MSEO transfers.

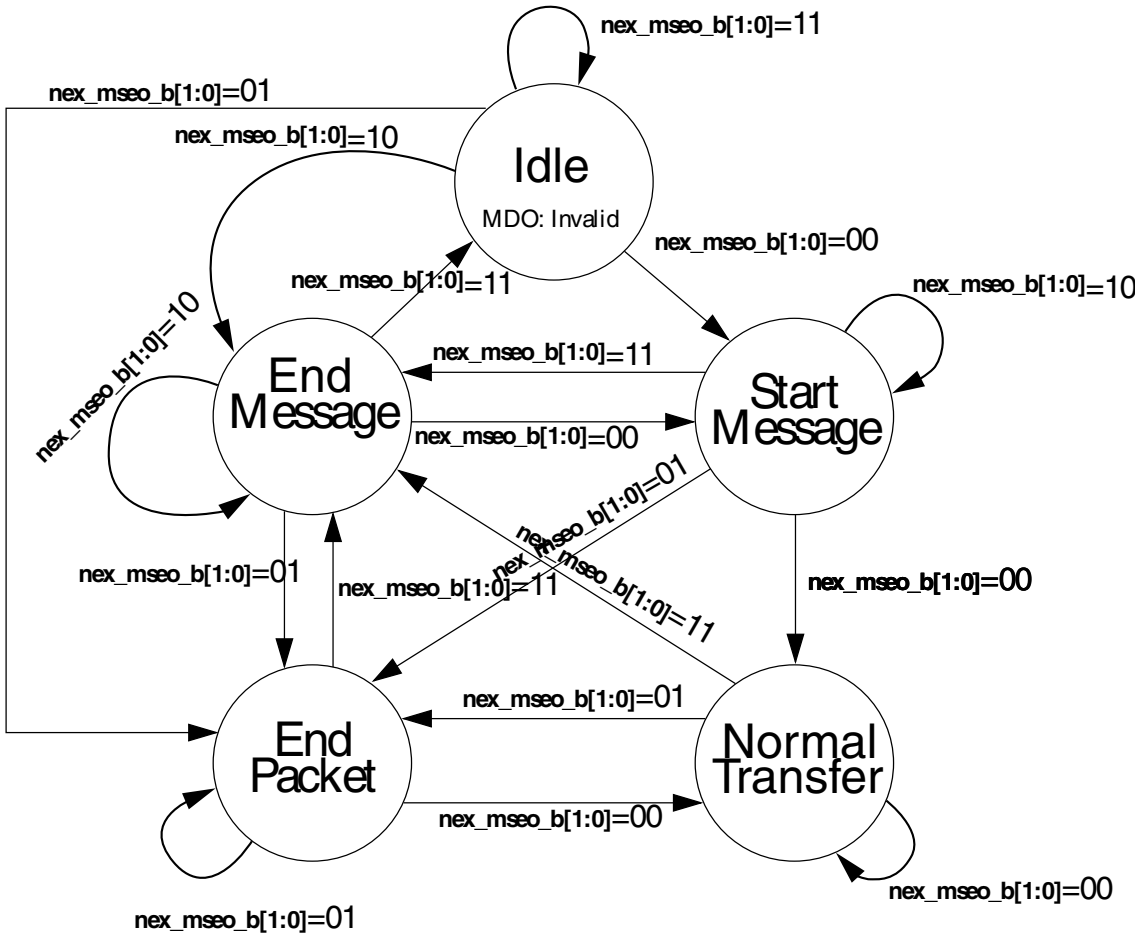


Figure 79-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

79.18 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

79.19 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 79-42 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 79-42. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

79.20 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 79-43 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 79-44 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 79-43. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 79-44. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A	S3	S2	0	1	End Packet

Table continues on the next page...

Table 79-44. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
						P					
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 79-45 and Table 79-46 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 79-45. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 79-46. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 79-47 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 79-48 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 79-47. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 79-48. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

79.21 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the Data Sheet.

79.22 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

79.22.1 JTAG sequence for accessing internal Nexus registers

Table 79-49. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

79.22.2 JTAG sequence for read access of memory-mapped resources

Table 79-50. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

79.22.3 JTAG sequence for write access of memory-mapped resources

Table 79-51. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** pin
8	—	If CNT > 0, go back to Step #5

Chapter 80
Nexus Module

80.1 Introduction

The e200z710n3Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

80.1.1 General description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

80.1.2 Terms and definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 80-1. Terms and definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

80.1.3 Feature list

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
    - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
    - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
    - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
    - One (1) Read/Write Error pin (**nex_err_b**) pin
    - One (1) Watchpoint Event output pin (**evto_b**)
    - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
    - One (1) Event In pin (**nex_evti_b**)
    - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)
- Indicates to the FCCU whether it becomes active during functional mode.

Note

For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (at the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

80.1.4 Functional block diagram

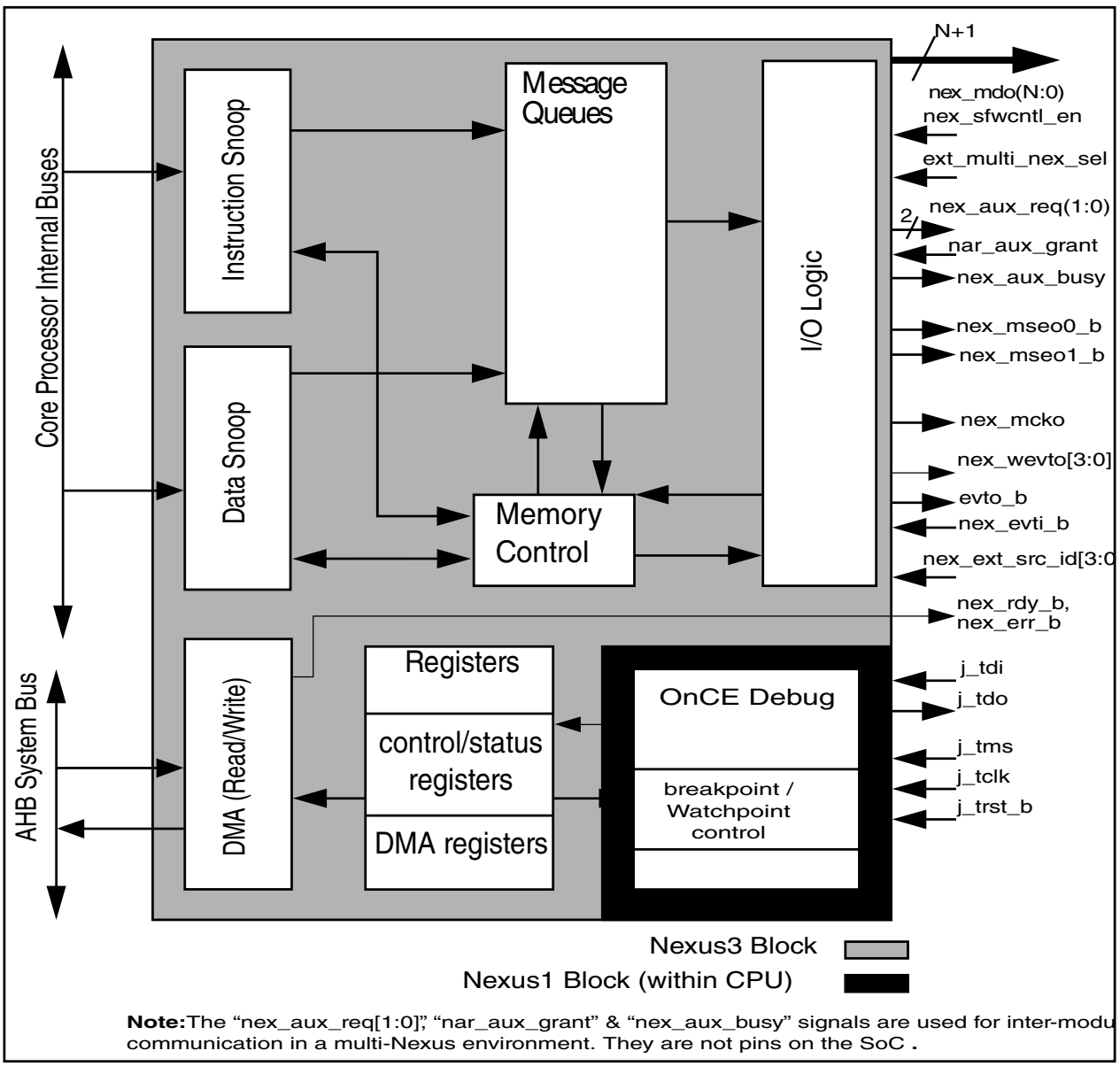


Figure 80-1. Nexus 3 functional block diagram

80.2 Enabling Nexus 3 operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

80.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

80.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 80-2.

Table 80-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 80-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table continues on the next page...

Table 80-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 80-7)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)

Table continues on the next page...

Table 80-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 80-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 80-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 80-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions

Table continues on the next page...

Table 80-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 80-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 80-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 80-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 80-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 80-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 80-5 shows the encodings used for resource codes for certain messages.

Table 80-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 80-6 shows the event code encodings used for certain messages.

Table 80-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace

Table continues on the next page...

Table 80-6. Event Code (EVCODE) Encoding (TCODE = 33) (continued)

Event Code	Description
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 80-7 shows the data trace size encodings used for certain messages.

Table 80-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

80.4 Nexus 3 Programmer's model

This section describes the Nexus 3 programmer's model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 80-8. Nexus 3 register map

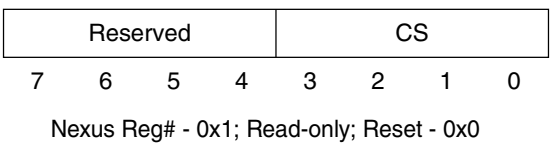
Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)[2]	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.

3. The "PCR_INDEX" is a parameter determined by the SoC.

80.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# - 0x1; Read-only; Reset - 0x0

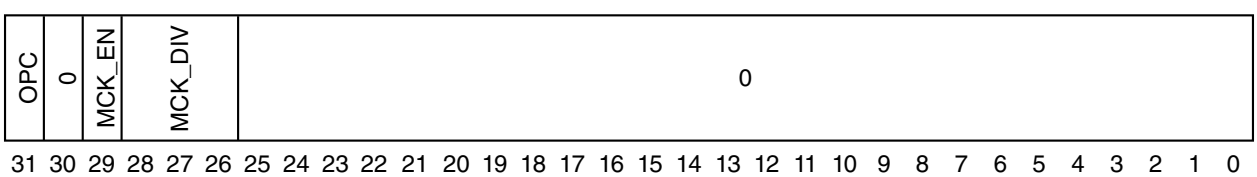
Figure 80-2. Client Select Control (CSC) register

Table 80-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

80.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.



Nexus Reg# - PCR_INDEX; Read/Write; Reset - 0x0

Figure 80-3. Port Configuration Register

Table 80-10. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality

Table continues on the next page...

Table 80-10. PCR field descriptions (continued)

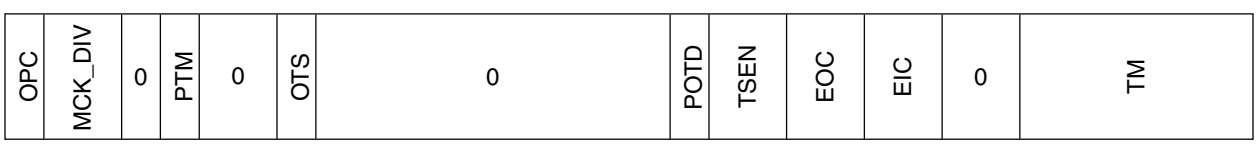
Bit	Name	Description
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

80.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 80-4 and its fields are described in Table 80-11.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 80-4. Development Control Register 1

Table 80-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1] 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering. Refer to External Hardware Trigger Controls for more information on external hardware triggering.
1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering. Refer to External Hardware Trigger Controls for more information on external hardware triggering.
1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering. Refer to External Hardware Trigger Controls for more information on external hardware triggering.

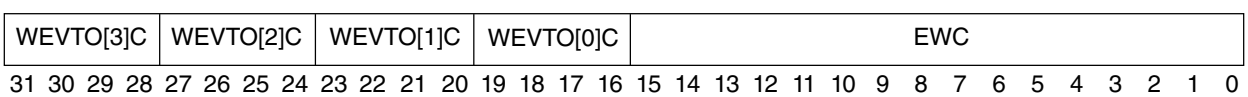
Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

80.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z710n3) Core Debug Support chapter.

Development Control Register 2 is shown in Figure 80-5 and its fields are described in Table 80-12.



Nexus Reg# - 0x3; DCR - 369; Read/Write; Reset - 0x0

Figure 80-5. Development Control Register 2

Table 80-12. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[3]** 0001 Watchpoint #0 triggers **nex_wevto[3]** 0010 Watchpoint #1 triggers **nex_wevto[3]**

Table continues on the next page...

Table 80-12. DC2 field descriptions (continued)

Bits	Name	Description
		0011 Watchpoint #2 triggers **nex_wevto[3]**
		0100 Watchpoint #3 triggers **nex_wevto[3]**
		0101 Watchpoint #4 triggers **nex_wevto[3]**
		0110 Watchpoint #5 triggers **nex_wevto[3]**
		0111 Watchpoint #6 triggers **nex_wevto[3]**
		1000 Watchpoint #7 triggers **nex_wevto[3]**
		1001 Watchpoint #8 triggers **nex_wevto[3]**
		1010 Watchpoint #9 triggers **nex_wevto[3]**
		1011 Watchpoint #10 triggers **nex_wevto[3]**
		1100 Watchpoint #11 triggers **nex_wevto[3]**
		1101 Watchpoint #12 triggers **nex_wevto[3]**
		1110 Watchpoint #13 triggers **nex_wevto[3]**
		1111 Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[2]**
		0001 Watchpoint #0 triggers **nex_wevto[2]**
		0010 Watchpoint #1 triggers **nex_wevto[2]**
		0011 Watchpoint #2 triggers **nex_wevto[2]**
		0100 Watchpoint #3 triggers **nex_wevto[2]**
		0101 Watchpoint #4 triggers **nex_wevto[2]**
		0110 Watchpoint #5 triggers **nex_wevto[2]**
		0111 Watchpoint #6 triggers **nex_wevto[2]**
		1000 Watchpoint #7 triggers **nex_wevto[2]**
		1001 Watchpoint #8 triggers **nex_wevto[2]**
		1010 Watchpoint #9 triggers **nex_wevto[2]**
		1011 Watchpoint #10 triggers **nex_wevto[2]**
		1100 Watchpoint #11 triggers **nex_wevto[2]**
		1101 Watchpoint #12 triggers **nex_wevto[2]**
		1110 Watchpoint #13 triggers **nex_wevto[2]**
		1111 Watchpoint #14 triggers **nex_wevto[2]**
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[1]**
		0001 Watchpoint #0 triggers **nex_wevto[1]**
		0010 Watchpoint #1 triggers **nex_wevto[1]**
		0011 Watchpoint #2 triggers **nex_wevto[1]**
		0100 Watchpoint #3 triggers **nex_wevto[1]**
		0101 Watchpoint #4 triggers **nex_wevto[1]**

Table continues on the next page...

Table 80-12. DC2 field descriptions (continued)

Bits	Name	Description
		0110 Watchpoint #5 triggers **nex_wevto[1]**
		0111 Watchpoint #6 triggers **nex_wevto[1]**
		1000 Watchpoint #7 triggers **nex_wevto[1]**
		1001 Watchpoint #8 triggers **nex_wevto[1]**
		1010 Watchpoint #9 triggers **nex_wevto[1]**
		1011 Watchpoint #10 triggers **nex_wevto[1]**
		1100 Watchpoint #11 triggers **nex_wevto[1]**
		1101 Watchpoint #12 triggers **nex_wevto[1]**
		1110 Watchpoint #13 triggers **nex_wevto[1]**
		1111 Watchpoint #14 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[0]**
		0001 Watchpoint #0 triggers **nex_wevto[0]**
		0010 Watchpoint #1 triggers **nex_wevto[0]**
		0011 Watchpoint #2 triggers **nex_wevto[0]**
		0100 Watchpoint #3 triggers **nex_wevto[0]**
		0101 Watchpoint #4 triggers **nex_wevto[0]**
		0110 Watchpoint #5 triggers **nex_wevto[0]**
		0111 Watchpoint #6 triggers **nex_wevto[0]**
		1000 Watchpoint #7 triggers **nex_wevto[0]**
		1001 Watchpoint #8 triggers **nex_wevto[0]**
		1010 Watchpoint #9 triggers **nex_wevto[0]**
		1011 Watchpoint #10 triggers **nex_wevto[0]**
		1100 Watchpoint #11 triggers **nex_wevto[0]**
		1101 Watchpoint #12 triggers **nex_wevto[0]**
		1110 Watchpoint #13 triggers **nex_wevto[0]**
		1111 Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration[1]
		0000000000000000 No Watchpoints #0-15 trigger **evto_b**
		XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **evto_b**
		XXXXXXXXXXXXXX1X Watchpoint #1 triggers **evto_b**
		XXXXXXXXXXXXX1XX Watchpoint #2 triggers **evto_b**
		XXXXXXXXXXXX1XXX Watchpoint #3 triggers **evto_b**
		XXXXXXXXXXX1XXXX Watchpoint #4 triggers **evto_b**
		XXXXXXXXXX1XXXXX Watchpoint #5 triggers **evto_b**
		XXXXXXXXX1XXXXXX Watchpoint #6 triggers **evto_b**
		XXXXXXXX1XXXXXXX Watchpoint #7 triggers **evto_b**

Table 80-12. DC2 field descriptions

Bits	Name	Description
		XXXXXXX1XXXXXXXX Watchpoint #8 triggers **evto_b**
		XXXXXX1XXXXXXXXX Watchpoint #9 triggers **evto_b**
		XXXXX1XXXXXXXXXX Watchpoint #10 triggers **evto_b**
		XXXX1XXXXXXXXXXX Watchpoint #11 triggers **evto_b**
		XXX1XXXXXXXXXXXX Watchpoint #12 triggers **evto_b**
		XX1XXXXXXXXXXXXX Watchpoint #13 triggers **evto_b**
		X1XXXXXXXXXXXXXX Watchpoint #14 triggers **evto_b**
		1XXXXXXXXXXXXXXX Watchpoint #15 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 80-6 and its fields are described in Table 80-13.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 80-6. Development Control 3 (DCR3) register

Table 80-13. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration
		0000 No Watchpoints #15-#26 trigger **nex_wevto[3]**
		0001 Watchpoint #15 triggers **nex_wevto[3]**
		0010 Watchpoint #16 triggers **nex_wevto[3]**
		0011 Watchpoint #17 triggers **nex_wevto[3]**
		0100 Watchpoint #18 triggers **nex_wevto[3]**
		0101 Watchpoint #19 triggers **nex_wevto[3]**
		0110 Watchpoint #20 triggers **nex_wevto[3]**
		0111 Watchpoint #21 triggers **nex_wevto[3]**
		1000 Watchpoint #22 triggers **nex_wevto[3]**
		1001 Watchpoint #23 triggers **nex_wevto[3]**
		1010 Watchpoint #24 triggers **nex_wevto[3]**
		1011 Watchpoint #25 triggers **nex_wevto[3]**
		1100 Watchpoint #26 triggers **nex_wevto[3]**
		1101 - 1111 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration
		0000 No Watchpoints #15-#26 trigger **nex_wevto[2]**

Table continues on the next page...

Table 80-13. DC3 field descriptions (continued)

Bits	Name	Description
		0001 Watchpoint #15 triggers **nex_wevto[2]**
		0010 Watchpoint #16 triggers **nex_wevto[2]**
		0011 Watchpoint #17 triggers **nex_wevto[2]**
		0100 Watchpoint #18 triggers **nex_wevto[2]**
		0101 Watchpoint #19 triggers **nex_wevto[2]**
		0110 Watchpoint #20 triggers **nex_wevto[2]**
		0111 Watchpoint #21 triggers **nex_wevto[2]**
		1000 Watchpoint #22 triggers **nex_wevto[2]**
		1001 Watchpoint #23 triggers **nex_wevto[2]**
		1010 Watchpoint #24 triggers **nex_wevto[2]**
		1011 Watchpoint #25 triggers **nex_wevto[2]**
		1100 Watchpoint #26 triggers **nex_wevto[2]**
		1101 - 1111 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration
		0000 No Watchpoints #15-#29 trigger **nex_wevto[1]**
		0001 Watchpoint #15 triggers **nex_wevto[1]**
		0010 Watchpoint #16 triggers **nex_wevto[1]**
		0011 Watchpoint #17 triggers **nex_wevto[1]**
		0100 Watchpoint #18 triggers **nex_wevto[1]**
		0101 Watchpoint #19 triggers **nex_wevto[1]**
		0110 Watchpoint #20 triggers **nex_wevto[1]**
		0111 Watchpoint #21 triggers **nex_wevto[1]**
		1000 Watchpoint #22 triggers **nex_wevto[1]**
		1001 Watchpoint #23 triggers **nex_wevto[1]**
		1010 Watchpoint #24 triggers **nex_wevto[1]**
		1011 Watchpoint #25 triggers **nex_wevto[1]**
		1100 Watchpoint #26 triggers **nex_wevto[1]**
		1101 Watchpoint #27 triggers **nex_wevto[1]**
		1110 Watchpoint #28 triggers **nex_wevto[1]**
		1111 Watchpoint #29 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration
		0000 No Watchpoints #15-#29 trigger **nex_wevto[0]**
		0001 Watchpoint #15 triggers **nex_wevto[0]**
		0010 Watchpoint #16 triggers **nex_wevto[0]**
		0011 Watchpoint #17 triggers **nex_wevto[0]**
		0100 Watchpoint #18 triggers **nex_wevto[0]**
		0101 Watchpoint #19 triggers **nex_wevto[0]**

Table continues on the next page...

Table 80-13. DC3 field descriptions (continued)

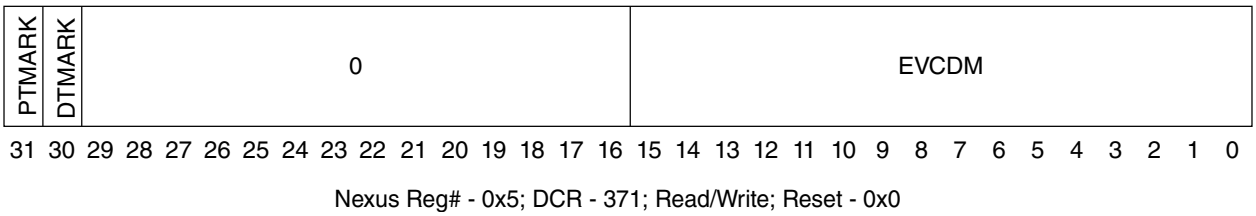
Bits	Name	Description
		0110 Watchpoint #20 triggers **nex_wevto[0]**
		0111 Watchpoint #21 triggers **nex_wevto[0]**
		1000 Watchpoint #22 triggers **nex_wevto[0]**
		1001 Watchpoint #23 triggers **nex_wevto[0]**
		1010 Watchpoint #24 triggers **nex_wevto[0]**
		1011 Watchpoint #25 triggers **nex_wevto[0]**
		1100 Watchpoint #26 triggers **nex_wevto[0]**
		1101 Watchpoint #27 triggers **nex_wevto[0]**
		1110 Watchpoint #28 triggers **nex_wevto[0]**
		1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration[1]
		00000000000000 No Watchpoints #16-#29 trigger **evto_b**
		XXXXXXXXXXXXX1 Watchpoint #16 triggers **evto_b**
		XXXXXXXXXXXX1X Watchpoint #17 triggers **evto_b**
		XXXXXXXXXXX1XX Watchpoint #18 triggers **evto_b**
		XXXXXXXXXX1XXX Watchpoint #19 triggers **evto_b**
		XXXXXXXXX1XXXX Watchpoint #20 triggers **evto_b**
		XXXXXXXX1XXXXX Watchpoint #21 triggers **evto_b**
		XXXXXXX1XXXXXX Watchpoint #22 triggers **evto_b**
		XXXXXX1XXXXXXX Watchpoint #23 triggers **evto_b**
		XXXXX1XXXXXXXX Watchpoint #24 triggers **evto_b**
		XXXX1XXXXXXXXX Watchpoint #25 triggers **evto_b**
		XXX1XXXXXXXXXX Watchpoint #26 triggers **evto_b**
		XX1XXXXXXXXXXX Watchpoint #27 triggers **evto_b**
		X1XXXXXXXXXXXX Watchpoint #28 triggers **evto_b**
		1XXXXXXXXXXXXX Watchpoint #29 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

80.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 80-7 and its fields are described in Table 80-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

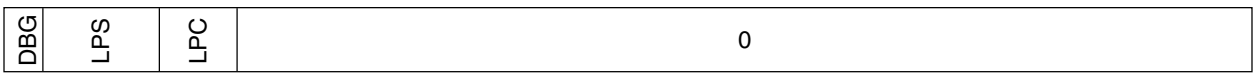
Figure 80-7. Development Control 4 (DC4) register

Table 80-14. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask For implemented EVCODEs. please see Table 80-6. 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages 1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

80.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 80-8. Development Status Register

Table 80-15. DS field description

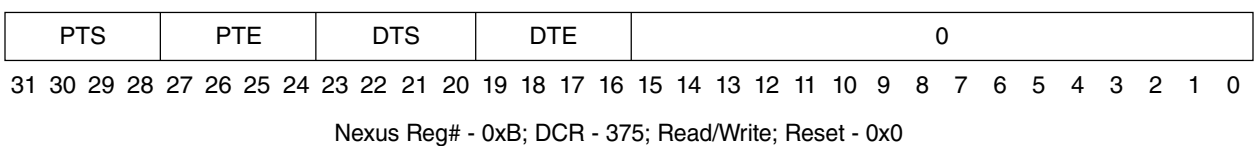
Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

80.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data

Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z710n3) Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 80-9. Watchpoint Trigger (WT) Register

Table 80-16 details the Watchpoint Trigger register fields.

Table 80-16. WT field descriptions

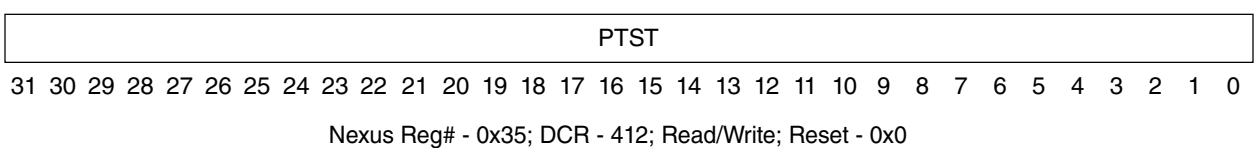
Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register

Table continues on the next page...

Table 80-16. WT field descriptions (continued)

Bits	Name	Description
27:24	PTE	PTE - Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.



Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 80-10. Program Trace Start Trigger Control (PTSTC) register

Table 80-17 details the PTSTC register fields.

Table 80-17. Program Trace Start Trigger Control Register Fields

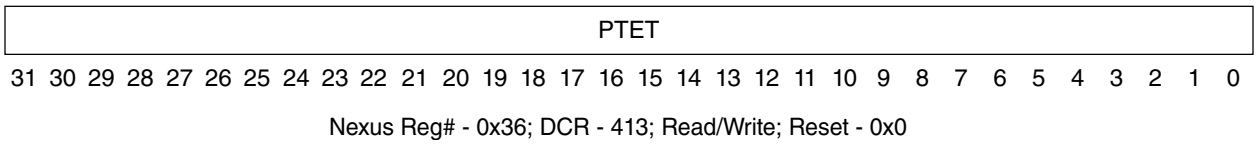
Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control

Table continues on the next page...

Table 80-17. Program Trace Start Trigger Control Register Fields (continued)

Bits	Name	Description
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.



Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 80-11. Program Trace End Trigger Control (PTETC) register

Table 80-18 details the PTETC register fields.

Table 80-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28

Table 80-18. PTETC field descriptions

Bits	Name	Description
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

	DTST

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 80-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 80-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18

Table 80-19. DTSTC field descriptions

Bits	Name	Description
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

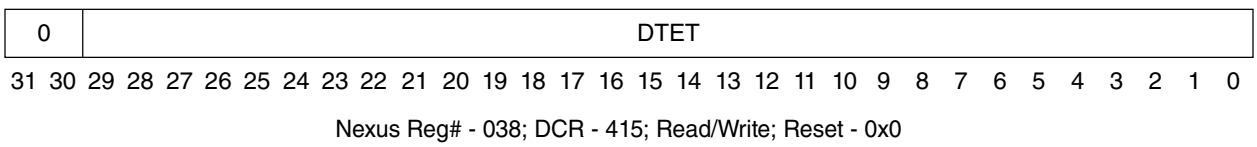


Figure 80-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 80-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control
		00000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10

Table 80-20. DTET field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		X1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

80.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 80-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 80-21. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging

Table continues on the next page...

Table 80-21. WMSK field descriptions (continued)

Bits	Name	Description
		00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

80.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

0	SPTHOLD	0	SPEN	0	STTHOLD	0	STEN

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 80-15. Nexus Overrun Control Register

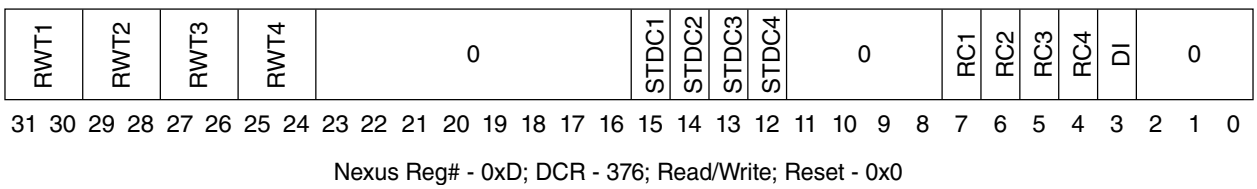
Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 80-16. Nexus Overrun Control Register Fields

80.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.



Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 80-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 80-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1

Table continues on the next page...

Table 80-22. DTC field descriptions (continued)

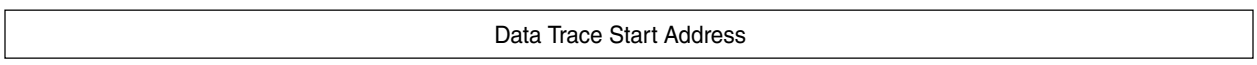
Bits	Name	Description
		1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core (e200z710n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

80.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

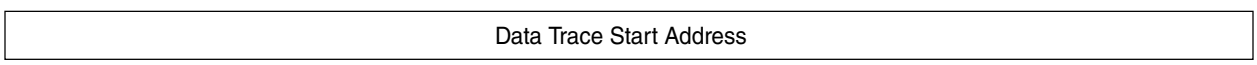
Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 80-18. Data Trace Start Address 1 (DTSA1) register



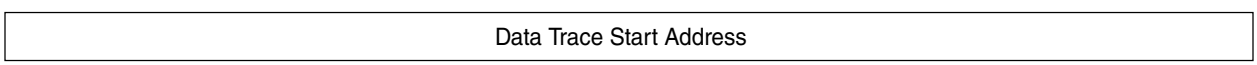
Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 80-19. Data Trace Start Address 2 (DTSA2) register



Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 80-20. Data Trace Start Address 3 (DTSA3) register

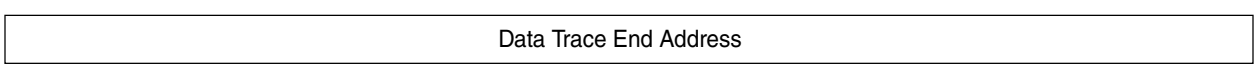


Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 80-21. Data Trace Start Address 4 (DTSA4) register

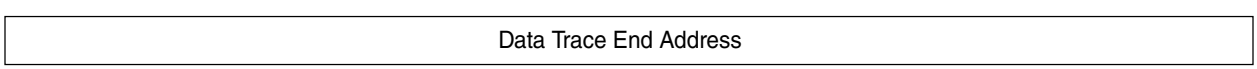
80.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.



Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 80-22. Data Trace End Address 1 (DTEA1) register



Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 80-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 80-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 80-25. Data Trace End Address 4 (DTEA4) register

Table 80-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 80-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

80.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 80-25.

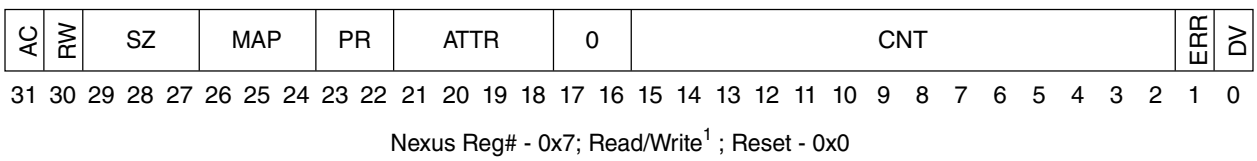


Figure 80-26. Read/Write Access Control/Status (RWCS) register

NOTES:

[1]ERR and DV are read-only

Table 80-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority

Table continues on the next page...

Table 80-24. RWCS field description (continued)

Bits	Name	Description
		00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR[2]	Read/Write Access Error (see Table 80-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 80-25)

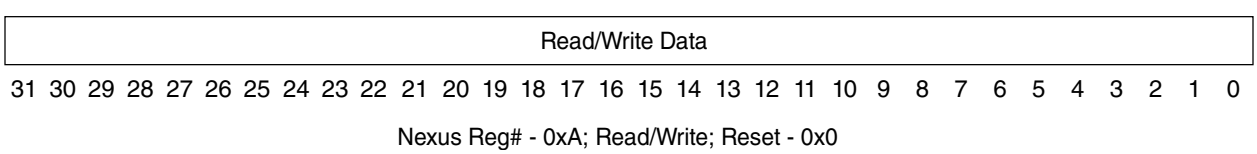
1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 80-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

80.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.



Nexus Reg# - 0xA; Read/Write; Reset - 0x0

Figure 80-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 80-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 80-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 80-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 80-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

80.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

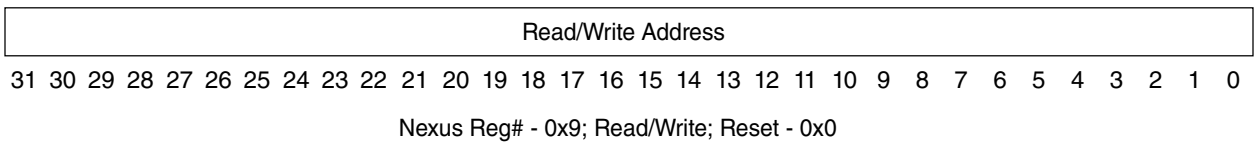


Figure 80-28. Read/Write Access Address (RWA) register

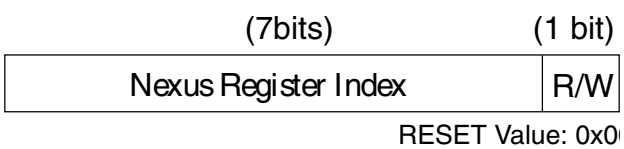
80.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 80-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 80-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:



Nexus Register Index	Selected from values in Table 80-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
    a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
    b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

80.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 80-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

80.7 Nexus message fields

Nexus messages are composed of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

80.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

80.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

80.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

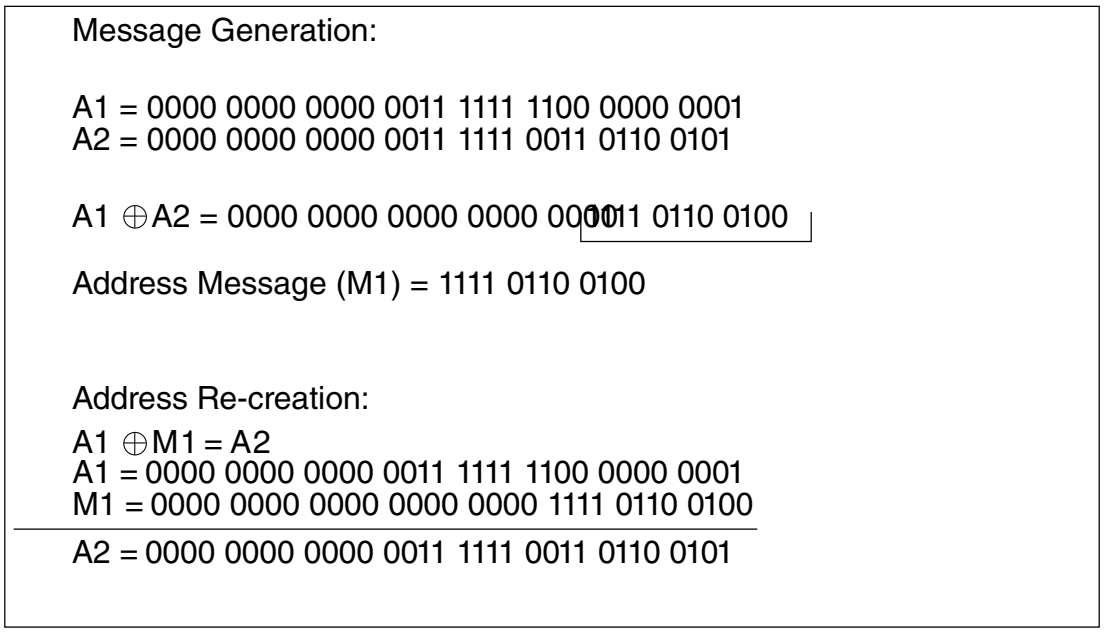


Figure 80-29. Relative address generation and recreation

80.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

80.7.5 Timestamp field (TSTAMP)

Timestamps may be optionally applied to messages. Enabling of timestamping is controlled by the nex_tsen input signal. The timestamping mode (timestamp frequency of occurrence) is controlled by the value of nex_tsmode[0:1] and the timestamp value is based on the nex_timebase[0:29] inputs. These signals are first synchronized to the nex_clk clock domain, and for nex_timebase[0:29], a gray-code to binary value

conversion is performed. The timestamp value is applied to messages as they exit the message queues. Leading zeros in the timestamp value are removed. shows the nex_tsmode[0:1] encodings used for the timestamp mode.

Table 80-28. nex_tsmode[0:1] timestamp mode encoding

nex_tsmode[0:1]	Timestamp Mode
00	Timestamp applied to every message
01	Timestamp applied to every 4th message
10	Timestamp applied to every 16th message
11	Timestamp applied to every 64th message

80.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

80.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

80.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

80.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

80.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 80-29 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 80-29.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 80-29. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A[1]
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

80.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

80.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

80.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

80.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

80.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

80.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

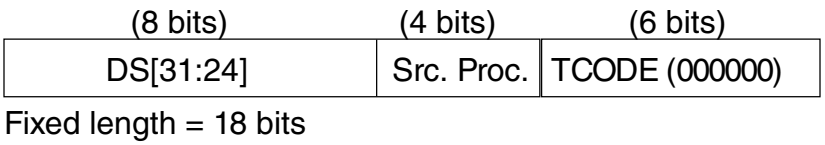


Figure 80-30. Debug Status message format

80.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 80-3 and Table 80-4):

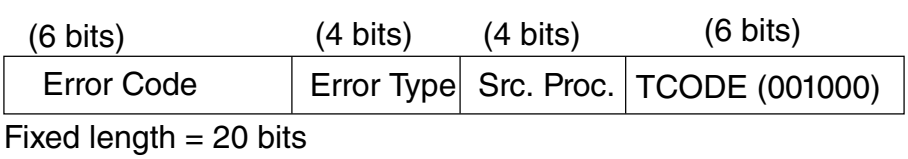


Figure 80-31. Error message format

80.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

80.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

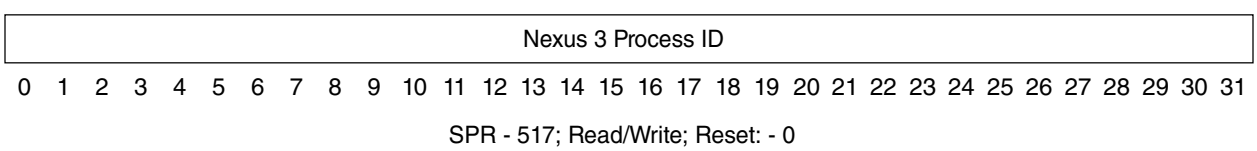
80.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z710n3) processors contain a *Power ISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.



SPR - 517; Read/Write; Reset: - 0

Figure 80-32. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 80-33. Ownership Trace Message format

80.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the e200z710n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for

Core (e200z710n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

80.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

80.12.1.1 Indirect Branch Message instructions

Table 80-30 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 80-30. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

80.12.1.2 Direct Branch Message Instructions

Table 80-31 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 80-31. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

80.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

80.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

80.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

80.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

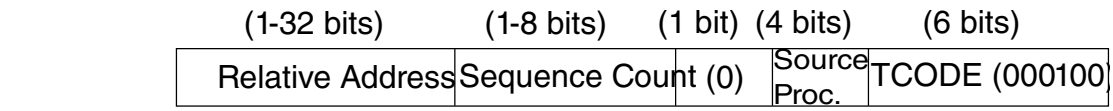
(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

Figure 80-34. Indirect branch message (history) format

80.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

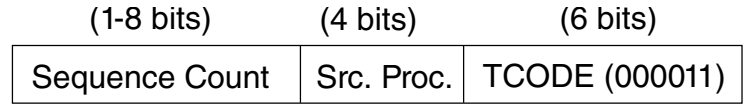


Max length = 51 bits; Min length = 13 bits

Figure 80-35. Indirect branch message format

80.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:



Max length = 18 bits; Min length = 11bits

Figure 80-36. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

80.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

80.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

80.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 80-32. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

80.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

80.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

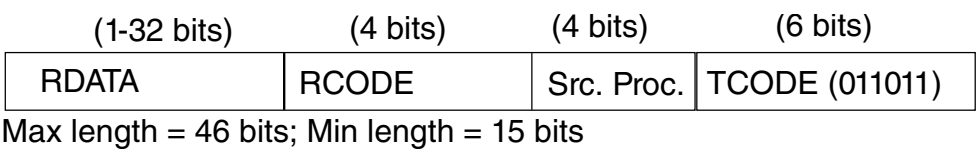


Figure 80-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 80-33. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

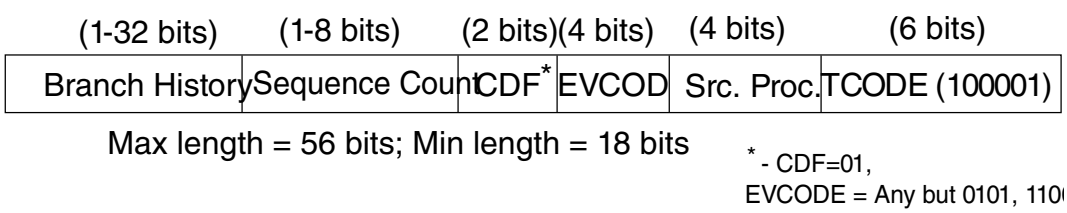
80.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 80-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:



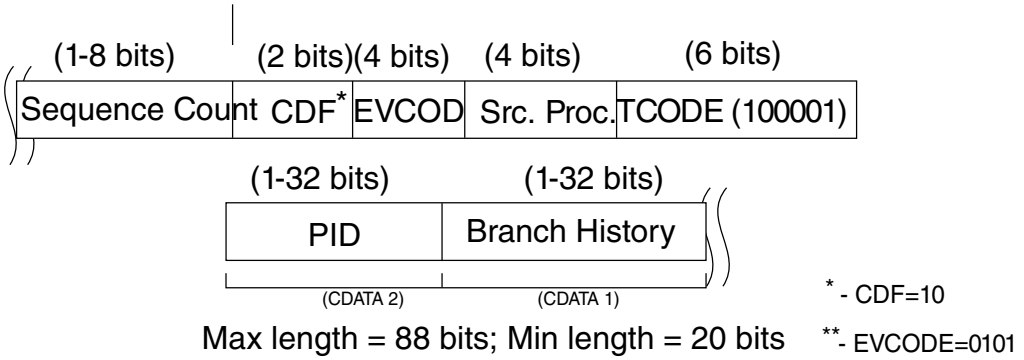


Figure 80-38. Program Correlation message formats

80.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

80.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

80.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 80-34):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in Table 80-34.

Table 80-34. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.

Table continues on the next page...

Table 80-34. Program Trace exception summary (continued)

Exception Condition	Exception Handling
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

80.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 80-34):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

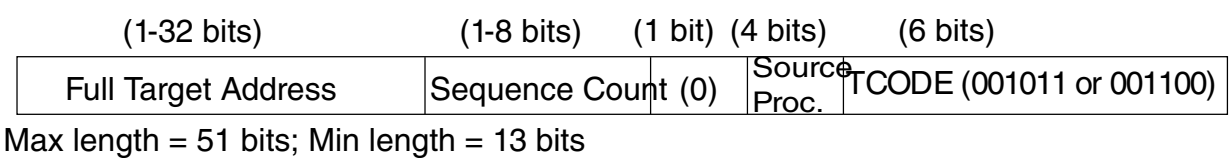
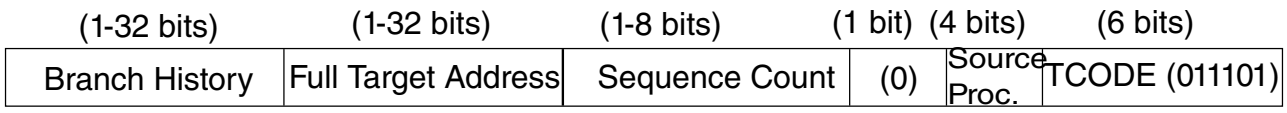


Figure 80-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:



Max length = 83 bits; Min length = 14 bits

Figure 80-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 80-35.

Table 80-35. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

80.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

80.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

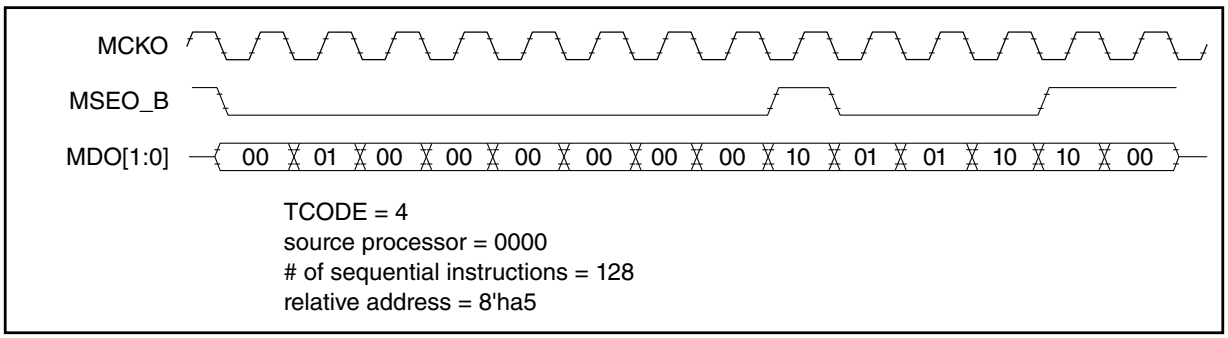


Figure 80-41. Program Trace - Indirect Branch message (Traditional)

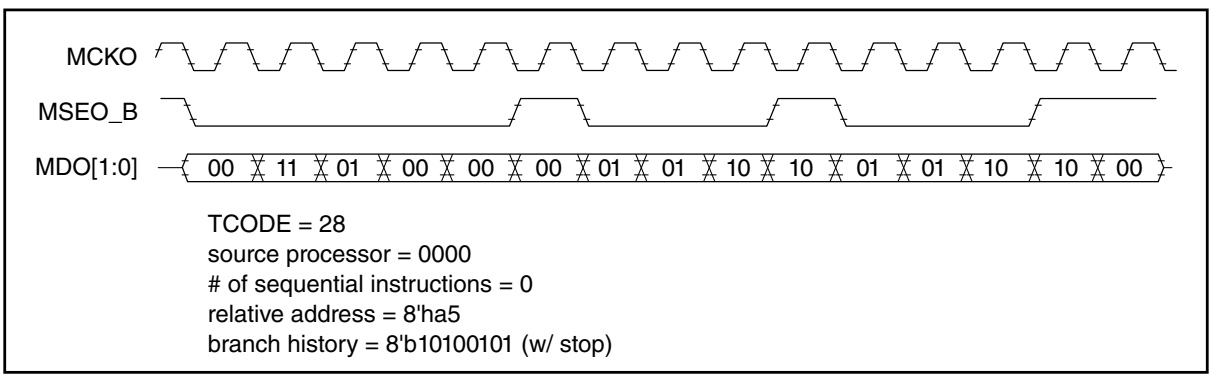


Figure 80-42. Program Trace - Indirect Branch message (history)

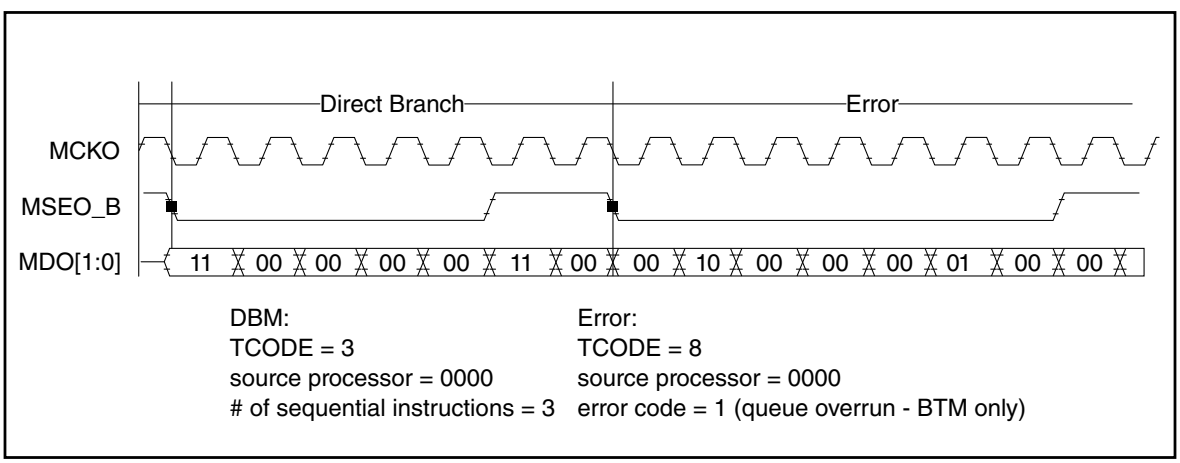


Figure 80-43. Program Trace - Direct Branch (traditional) & Error messages

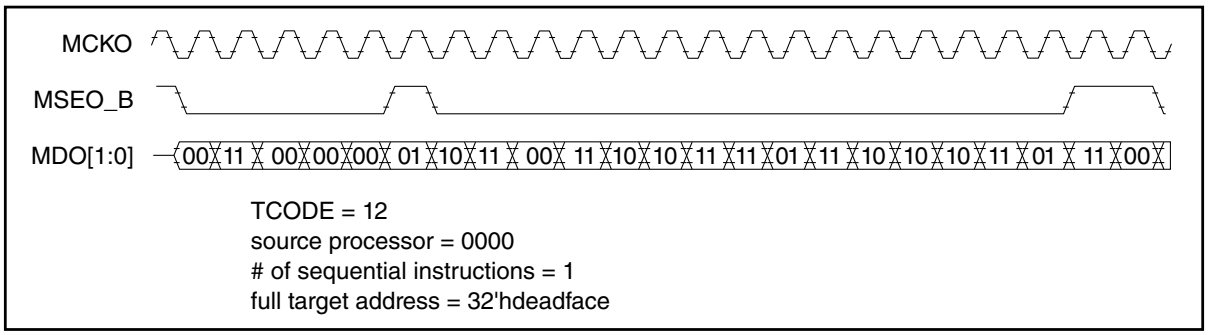


Figure 80-44. Program Trace - Indirect Branch w/ Sync. message

80.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

80.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z710n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of the following ways:

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

80.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

80.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 80-45. Data Write message format

80.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 80-46. Data Read fessage format

Note

* Core (e200z710n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z710n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

80.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 80-36):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

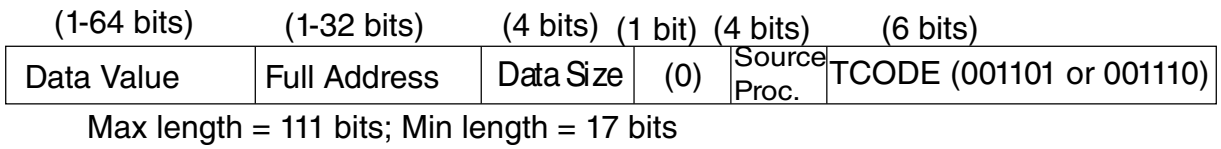


Figure 80-47. Data Write/Read with Sync. message format

Exception conditions that result in Data Trace Synchronization are summarized in Table 80-36.

Table 80-36. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

80.13.3 DTM Operation

80.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

80.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

80.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

80.13.3.4 Bus Cycle special cases

Table 80-37. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$), [1]	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the

doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z710n3) Core Debug Support chapter.)

Note

A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

80.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

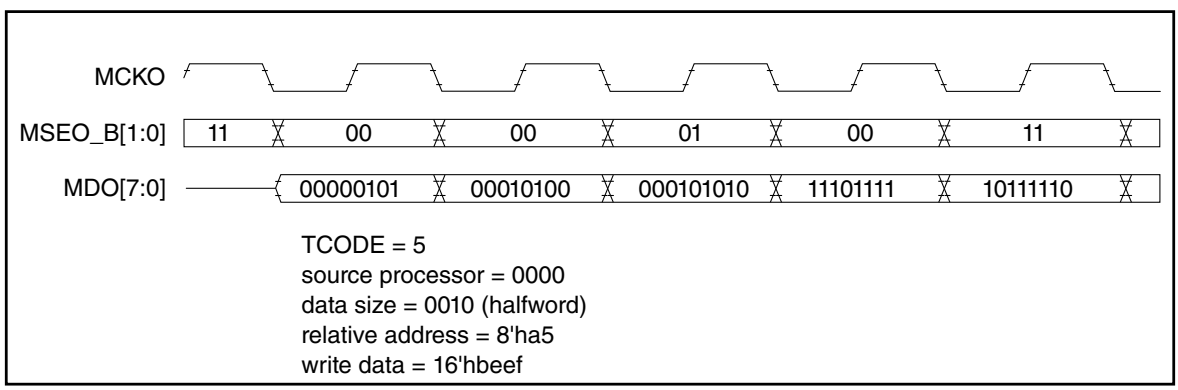


Figure 80-48. Data Trace - Data Write Message

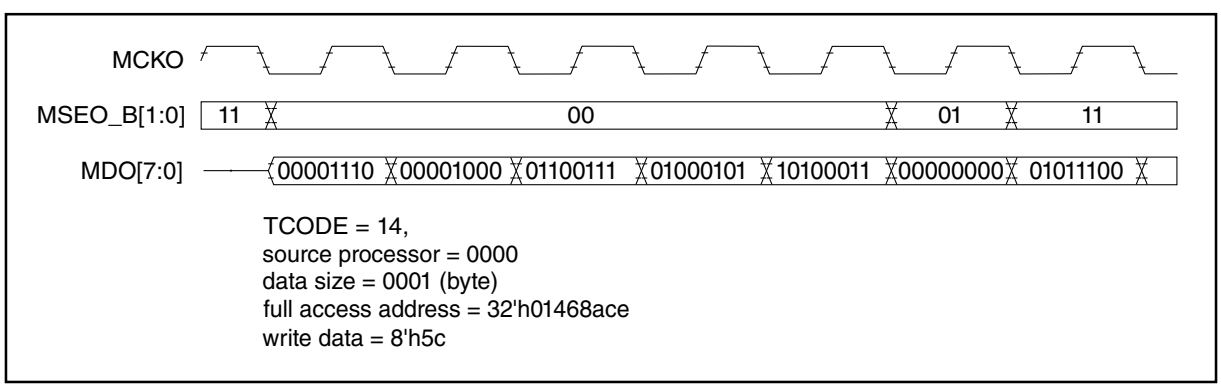


Figure 80-49. Data Trace — Data Read w/ Sync message

80.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

80.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

80.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

80.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core (e200z710n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

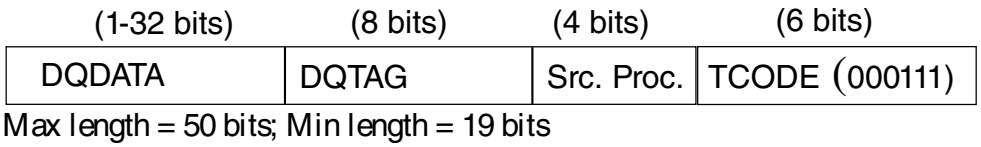


Figure 80-50. Data Acquisition message format

80.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z710n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 80-41 for details on **evto_b**.

Watchpoint information is messaged out in the following format:

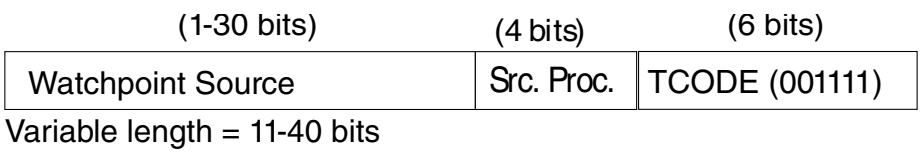


Figure 80-51. Watchpoint message format

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 80-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
000000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	

Table 80-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX	Watchpoint #3 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX	Watchpoint #4 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX	Watchpoint #5 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX	Watchpoint #6 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX	Watchpoint #7 enabled for WTM
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX	Watchpoint #8 enabled for WTM
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX	Watchpoint #9 enabled for WTM
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX	Watchpoint #10 enabled for WTM
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX	Watchpoint #11 enabled for WTM
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX	Watchpoint #12 enabled for WTM
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX	Watchpoint #13 enabled for WTM
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX	Watchpoint #14 enabled for WTM
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX	Watchpoint #15 enabled for WTM
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX	Watchpoint #16 enabled for WTM
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX	Watchpoint #17 enabled for WTM
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX	Watchpoint #18 enabled for WTM
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX	Watchpoint #19 enabled for WTM
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX	Watchpoint #20 enabled for WTM
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX	Watchpoint #21 enabled for WTM
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX	Watchpoint #22 enabled for WTM
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #23 enabled for WTM
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #24 enabled for WTM
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #25 enabled for WTM
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #26 enabled for WTM
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #27 enabled for WTM
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #28 enabled for WTM
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #29 enabled for WTM

80.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

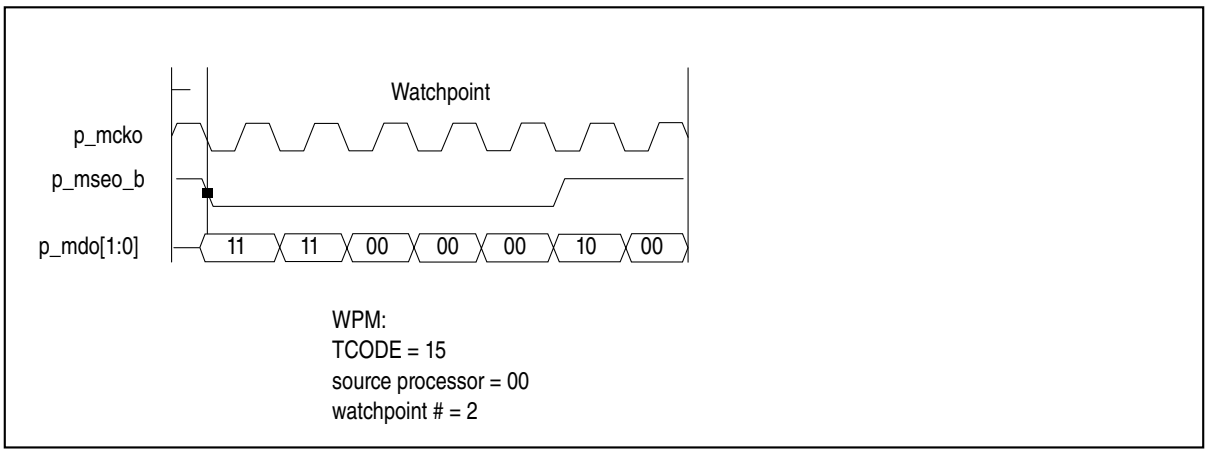


Figure 80-52. Watchpoint Message & Watchpoint Error message

80.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a "window" for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. The following table details the external hardware trigger signals.

Table 80-39. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0 → 1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

80.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/ OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

80.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'1 (write access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data (RWD) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)

4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD register to the memory mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 80-41 for details) and clears the ERR bit in the RWCS register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once the DV bit has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start**,**nex_err_b**, and **nex_rdy_b** pins, as well as the AC, DV, and ERR bits within the RWCS, provide Read/Write Access status to the external development tool.

80.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block then arbitrates for the AHB system bus and transfers the first data value from the RWD register to the memory-mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output is asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS register. Otherwise, the ERR bit is set and the DV bit is cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Clearing the DV bit indicates that the device is ready for the next access in the block transfer, or that an error has occurred.

3. If the AC bit has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS register are not changed when executing a block write access. The original values can be read by the external development tool at any time.

80.17.3 Single read access

1. Initialize the Read/Write Access Address (RWA)register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'0 (read access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

An Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 80-41 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin

will be asserted, the ERR bit will be set, and the DV bit will be cleared to indicate an error has occurred. The **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit is cleared in either case.

4. The data can then be read from the Read/Write Access Data register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start**, **nex_err_b**, and **nex_rdy_b** pins as well as the AC, DV, and ERR bits within the RWCS register provide Read/Write Access status to the external development tool.

80.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 80-41 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS register, and asserts the **nex_rdy_b** pin (see Table 80-41 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that

the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.

4. The data can then be read from the RWD register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

80.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

80.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write error

80.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
    a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
    b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

80.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

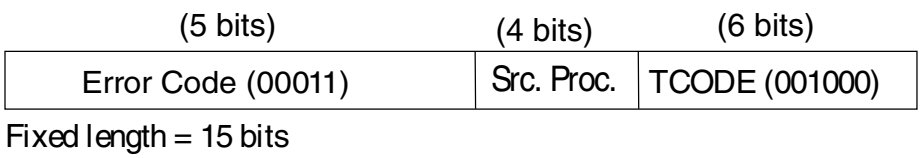


Figure 80-53. Error message format1

80.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

80.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 80-41. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core (e200z710n3) Core Debug Support chapter.

Table 80-40. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 80-41.

Table 80-41. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.

Table continues on the next page...

Table 80-41. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_ahb_start	O	AHB Start (**nex_ahb_start**) is an output pin used to indicate to the external tool or SoC that the Nexus block is requesting the next Read/Write Access on the system bus. If Nexus is enabled, this signal is asserted upon an acknowledged request to start an AHB system bus transfer (Nexus read or write) and is pulsed asserted for one **nex_clk** clock period, corresponding to the first clock of the address phase of the transfer. Upon exit from system reset or if Nexus is disabled, **nex_ahb_start** remains de-asserted.
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted.
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted.
evto_b	O	Event Out (**evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 80-42. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 80-44.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

80.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 80-43. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the .

Table 80-43. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 80-54 illustrates the state diagram for single pin MSEO transfers.

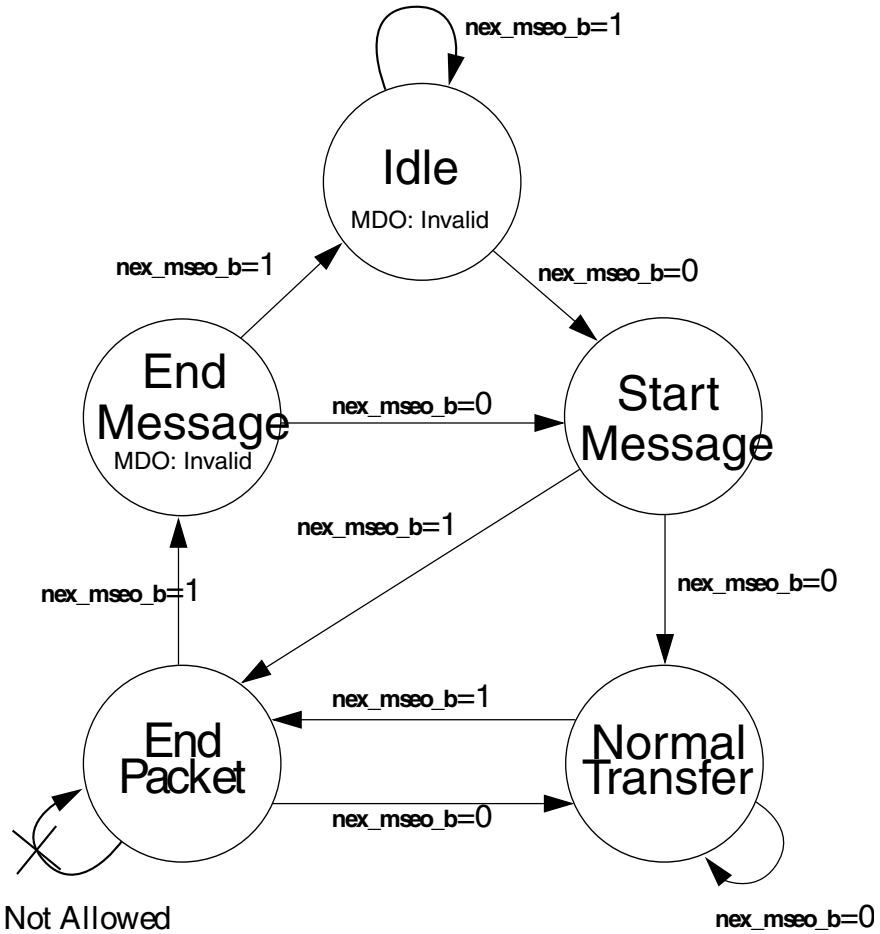


Figure 80-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 80-55 illustrates the state diagram for dual pin MSEO transfers.

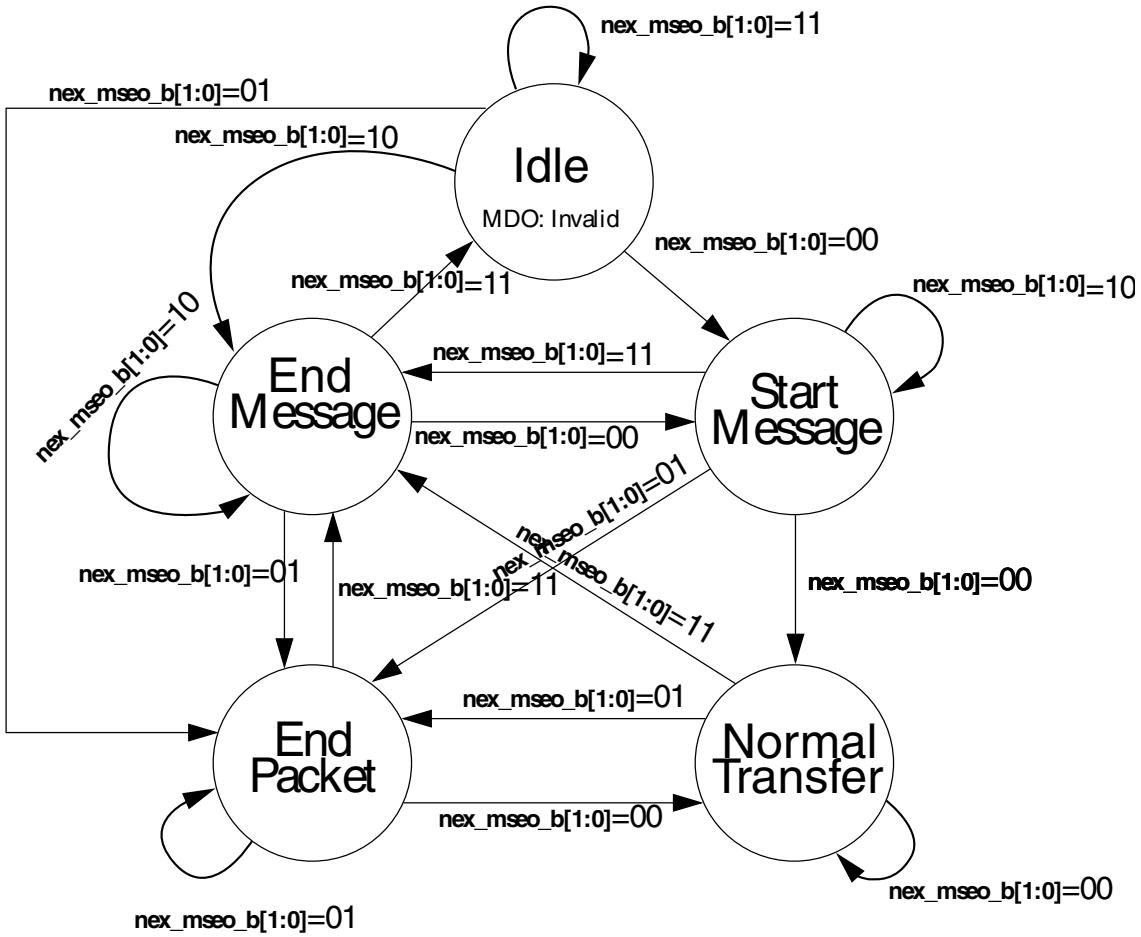


Figure 80-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

80.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

80.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 80-44 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 80-44. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

80.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 80-45 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 80-46 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 80-45. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 80-46. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A	S3	S2	0	1	End Packet

Table continues on the next page...

Table 80-46. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
						P					
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 80-47 and Table 80-48 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 80-47. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 80-48. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 80-49 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 80-50 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 80-49. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 80-50. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

80.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the Data Sheet.

80.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

80.23.1 JTAG sequence for accessing internal Nexus registers

Table 80-51. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

80.23.2 JTAG sequence for read access of memory-mapped resources

Table 80-52. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

80.23.3 JTAG sequence for write access of memory-mapped resources

Table 80-53. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b**or **nex_err_b**pin
8	—	If CNT > 0, go back to Step #5

Chapter 81
Nexus Crossbar Multi-Master Client (NXMC)

81.1 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools. This also provides the capability of sending user defined messages for the peripherals attached to it.

81.1.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- HSM master(s) filtering from the tracing function based on external control
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Peripheral interface for data message transfers based on In Circuit Trace Message (ICTM) format
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on external triggers.

- Optional timestamping of the messages based on timer value and controls received at the interface
- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

81.2 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NAR. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in Table 81-2.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see IEEE 1149.1 (JTAG) Test Access Port for details). A separate JTAG TAP controller is instantiated within the NXMC. The following table describes the JTAG pins.

Table 81-1. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 81-2. NXMC auxiliary pins

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: • one of the internal or external watchpoints has occurred and EOC = 2'b00 • system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see Enabling NXMC operation), assertion initiates data trace synchronization messages (provided data trace is enabled).

81.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in the following table. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 81-3. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 81-4)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 81-4)

Table continues on the next page...

Table 81-3. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 81-5)
	14	14	ECODE	fixed	Error code (See Table 81-6)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 81-4)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 81-4)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See Table 81-22)
	1	26	TSTAMP	variable	Timestamp (Optional)
Peripheral Message (In circuit Trace Message: ICTM	6	6	TCODE	fixed	TCODE number = 34
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	6	6	CKSRC	fixed	Peripheral source identifier

Table continues on the next page...

Table 81-3. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
	1	44	CKDATA	variable	Peripheral data
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 81-4. Data trace size (DSZ) encodings (TCODE = 58, 59, 60, 61)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 81-5. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 81-6. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved

Table continues on the next page...

Table 81-6. Error Code (ECODE) encodings (applicable based on error source) (continued)

ECODE		Description[1]
13 to 8	7 to 0	
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	In-Circuit Trace Message(s) Lost
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

81.4 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See IEEE 1149.1 (JTAG) Test Access Port for details. The complete list of registers is shown in this table.

Table 81-7. NXMC registers

Nexus register	Nexus access opcode (hex)	Read/ write	Read address (hex)	Write address (hex)
Development Control 1 (DC1)	2	R/W	04	05
Development Control 2 (DC2)	3	R/W	06	07
Watchpoint Trigger (WT)	B	R/W	16	17
Data Trace Control (DTC)	D	R/W	1A	1B
Data Trace Start Address1 (DTSA1)	E	R/W	1C	1D
Data Trace Start Address2 (DTSA2)	F	R/W	1E	1F
Data Trace End Address1 (DTEA1)	12	R/W	24	25
Data Trace End Address2 (DTEA2)	13	R/W	26	27
Breakpoint/Watchpoint Control Register1 (BWC1)	16	R/W	2C	2D
Breakpoint/Watchpoint Control Register2 (BWC2)	17	R/W	2E	2F
Breakpoint/Watchpoint Address Register1 (BWA1)	1E	R/W	3C	3D
Breakpoint/Watchpoint Address Register2 (BWA2)	1F	R/W	3E	3F

81.4.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	EOC		0	0	WEN	0	0	0	0	0	0	0	0	0	C	0	0	0	0	0	0	0	0	0	EIC		TM		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC1 is described in this table.

Table 81-8. DC1 register field description

Field	Description
31:29	Reserved for future functionality (read as 0)
28:27 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal or external) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
26:25	Reserved for future functionality (read as 0)
24 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
23:5	Reserved for future functionality (read as 0)
4:3 EIC	$\overline{\text{EVTI}}$ Control. 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
2:0 TM	Trace Mode. 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

81.4.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PME0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC2 is described in this table.

Table 81-9. DC2 register field description

Field	Description	
31:24	$\overline{\text{EVTO}}$ Watchpoint Configuration[1].	
EWC	00000000	No Watchpoints trigger $\overline{\text{EVTO}}$
	1XXXXXXX	External Watchpoint #1 triggers $\overline{\text{EVTO}}$
	X1XXXXXX	External Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XX1XXXXX	Reserved
	XXX1XXXX	Reserved
	XXXX1XXX	Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$
	XXXXX1XX	Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XXXXXX1X	Reserved
	XXXXXXX1	Reserved
23:1	Reserved for future functionality (read as 0)	
0	Peripheral Messaging Enable for the In-Circuit Trace (ICT) feature. [2]	
PME0	0	Peripheral Interface disabled
	1	Peripheral Interface enabled

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.
2. The peripheral details are described in the device-specific chapter that describes how the modules are configured.

81.4.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable.

The WT bits can be used to produce an address related window for triggering trace messages. This table shows the format of the WT register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The WT register is described in this table.

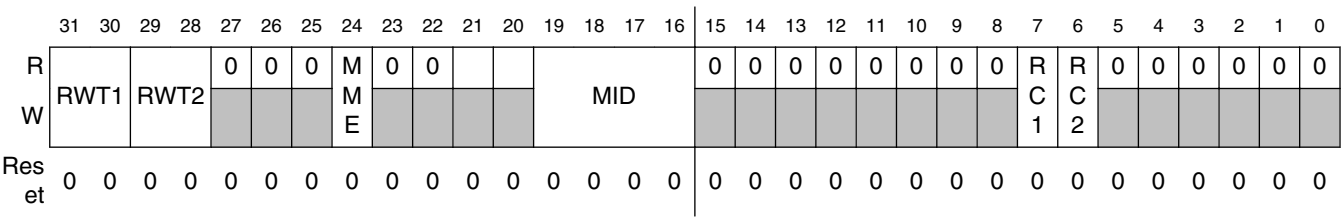
Table 81-10. WT register field description

Field	Description[1]
31:26	Reserved for future functionality (read as 0)
25:23 DTS	Data Trace Start Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
22:20 DTE	Data Trace End Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
19:0	Reserved for future functionality (read as 0)

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

81.4.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. This table shows the format of the DTC register.



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RWT1		RWT2		0	0	0	MME	0	0			MID				0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DTC register is described in this table.

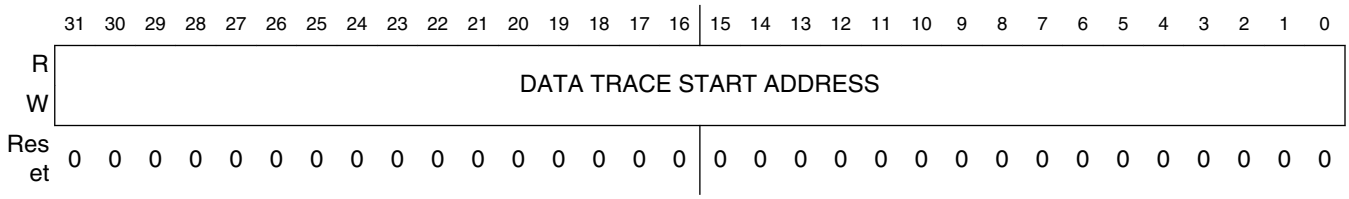
Table 81-11. DTC field description

Field	Description
31:30 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
29:28 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
27:25	Reserved for future functionality
24 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
23:20	Reserved for future functionality
19:16 MID	ID of the system bus Master to be traced[1]
15:8	Reserved for future functionality
7 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5:0	Reserved for future functionality

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

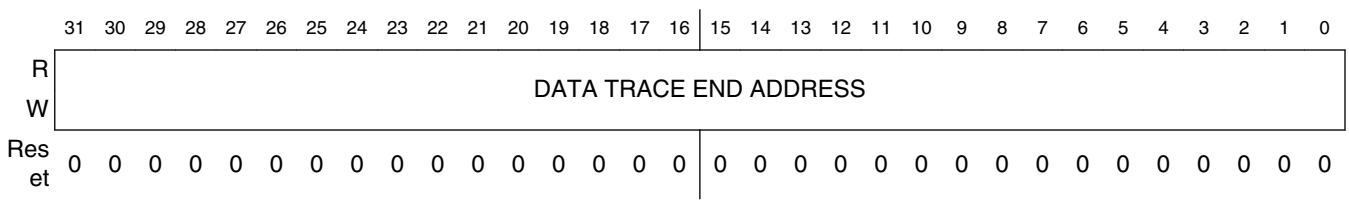
81.4.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. This table shows the format of the DTSA1 and DTSA2 registers.



81.4.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. This table shows the format of the DTEA1 and DTEA2 registers.



This table illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 81-12. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace

Note

DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

81.4.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. This table shows the format of the BWC1 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1		BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC1 register is described in this table.

Table 81-13. BWC1 field description

Field	Description	
31:30	Breakpoint/Watchpoint #1 Enable.	
BWE1	00	Internal Nexus Watchpoint #1 disabled
	01	Reserved
	10	Reserved
	11	Internal Nexus Watchpoint #1 enabled
29:28	Breakpoint/Watchpoint #1 Read/Write Select.	
BRW1	00	Watchpoint #1 hit on read accesses
	01	Watchpoint #1 hit on write accesses
	10	Watchpoint #1 on read or write accesses
	11	Reserved
27:18	Reserved for future functionality (read as 0)	
17:16	Breakpoint/Watchpoint #1 Register Compare.	
BWR1	00	No Register Compare (same as BWC1[31:30] = 2'b00)
	01	Reserved
	10	Compare with BWA1 value
	11	Reserved
15	Breakpoint/Watchpoint #1 Type.	
BWT1	0	Reserved
	1	Watchpoint #1 on data accesses
14:0	Reserved for future functionality (read as 0)	

81.4.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. This table shows the format of the BWC2 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2		BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC2 register is described in this table.

Table 81-14. BWC2 field description

Field	Description
31:30 BWE2	Breakpoint/Watchpoint #2 Enable. 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
29:28 BRW2	Breakpoint/Watchpoint #2 Read/Write Select. 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
15 BWT2	Breakpoint/Watchpoint #2 Type. 0 Reserved 1 Watchpoint #2 on data accesses
14:0	Reserved for future functionality (read as 0)

81.4.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. This table shows the format of the BWA1 and BWA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BREAKPOINT / WATCHPOINT ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

81.4.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 81-7. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

81.5 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion. The JTAG ID register should be read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

81.5.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in NXMC register access via JTAG.

81.5.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

81.5.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

81.5.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR).

Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 81-7.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 81-7), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown below.

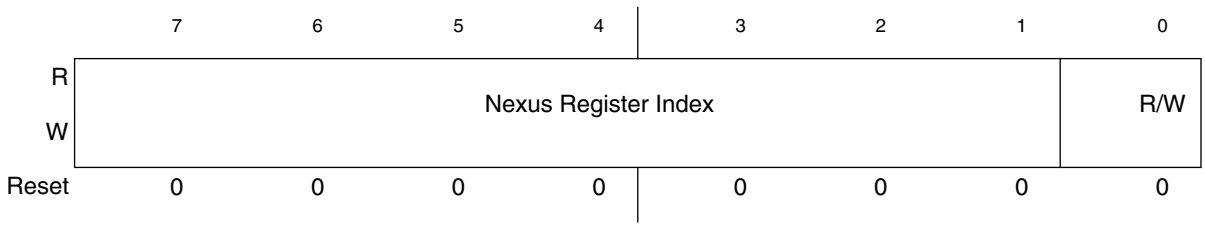


Table 81-15. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 81-7
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

81.6 Functional description

The following sections describe the functionality of the NXMC module.

81.6.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

81.6.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

81.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

81.6.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32 , 64. The TSTAMP field is 26 bits. Data write message and data read message information is messaged out in the format shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 81-16. DTM message format for DATA width = 64 bits

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

TSTAMP is another optional field which is inserted based on the controls received from the NAR, see Timestamping feature.

81.6.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

81.6.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in this figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 81-17. DTM error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

81.6.1.2.4 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 81-19):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.
- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.

- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature.
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message.This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in the following figure. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 81-18. Data write/read with synchronization message format

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Exception conditions that result in data trace synchronization are summarized in this table.

Table 81-19. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).

Table continues on the next page...

Table 81-19. Data trace exception summary (continued)

Exception condition	Exception handling
Collision Priority	All messages have the following priority: Error > WPM > ICTM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

81.6.1.3 Enabling data trace messaging operation

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

81.6.1.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note

If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > ICTM > DTM).

81.6.1.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in this figure.

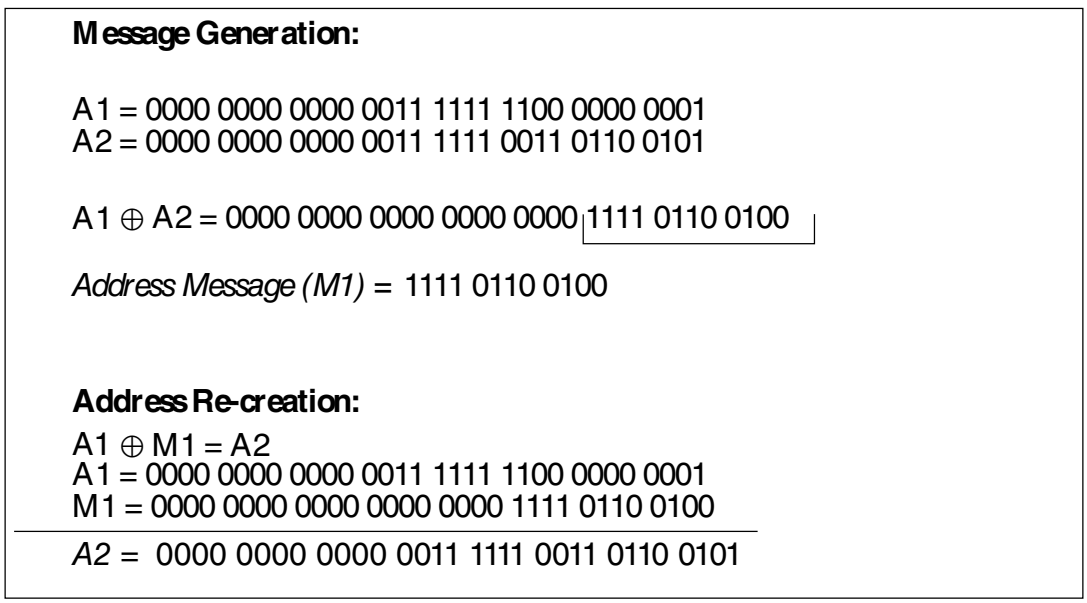


Figure 81-1. Relative addressing example

81.6.1.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

81.6.1.7 System bus cycle special cases

The system bus cycle special cases are shown in this table.

Table 81-20. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

81.6.1.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in Data write and data read messages for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

81.6.1.9 HSM master filtering

For security reasons, certain HSM masters need to be disabled or filtered from data tracing operations.

81.6.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

81.6.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints—Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in System bus multi-master tracing).
- External watchpoints—The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level. The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (i.e, on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in Table 81-21 and Table 81-22.

Table 81-21. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 81-22. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1[1]	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)

Table continues on the next page...

Table 81-22. Watchpoint source description (continued)

Watchpoint source (8-bits)	Watchpoint description
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

81.6.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in this figure.

Table 81-23. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

81.6.3 Peripheral interface

The NXMC provides an asynchronous interface to peripherals for sending data messages. The NXMC can back pressure or delay the data transfer using a data accept signal. A round-robin scheme is followed between the peripherals for forwarding received data to the queue; therefore, each peripheral has an equal priority and hence bandwidth for data transfers.

To enable messaging from peripherals there are controls present in the DC2 register for individual peripherals: DC2[PME0]. There are no message generated and hence data accepted from a peripheral unless the corresponding enable bit is set to 1.

Table 81-24. Peripheral DATA bus fields

MSB	6 bits	(54-6) bits			LSB
CKSRCn		DATAn			
54-1	54-6	54-6-1	2	1	0

n = peripheral number

Another input port is present which indicates data loss inside the peripheral itself due to over-run, the module in turn generates error message to communicate such events to the debugger.

81.6.3.1 Peripheral data message format (ICTM)

The message formats for the peripheral data messages (formatted as ICTM) is shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured. The CKSRC field is received from the MSB 6 bits of the individual peripheral data bus, and CKDATA is a variable field and contains 44 LSBits received on the databus of the peripheral interface.

Table 81-25. Peripheral message format (max length = 60 bits, min length = 17 bits)

1–44 bits	6 bits	4 bits	6 bits
CKDATAn	CKSRCn	SRC	TCODE (100010)

81.6.3.2 Peripheral error message format

The error messages are generated based data over-run indication from a peripheral. Error information is messaged out in the format shown in this figure.

Table 81-26. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE(CKSRCn,ErrCODE)	ETYPE	SRC	TCODE (001000)

The SRC field values are described in the device-specific chapter that describes how the modules are configured. ETYPE and ECODE encoding are listed in Table 81-5 and Table 81-6.

81.6.4 Timestamping feature

This feature allows a timestamp packet to be appended to all messages, based on the timer value and controls received externally from the NAR. If enabled, this packet is added as the last field to all the outgoing messages. The TSTAMP packet width is 26 bits.

The TSTAMP packet is appended at the end of a message during message formatting, before transfer. This figure shows an example of ICTM message with a TSTAMP field.

Table 81-27. Message with TSTAMP field example

26 bits	1–44 bits	6 bits	4 bits	6 bits
TSTAMP	CKDATAn	CKSRCn	SRC	TCODE (100010)

The enable or disable indication should remain static for the complete debug session. Dynamically enabling or disabling of the timestamping is not allowed during running debug tracing. The following steps allow seamless enabling/disabling of the timestamping:

1. All tracing (DTM/WPM/ICTM) must be disabled
2. Wait for the internal queues to be emptied (no further request is asserted at the auxiliary interface)
3. Change the timestamp enable indication to assert/deassert
4. Enable the required tracing controls (via TM, PME0 etc. control bits)

Chapter 82
Cyclic Redundancy Check (CRC) Unit

82.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

82.2 Main features

- Single context is supported
- Zero-wait states during the CRC computation (pipeline scheme)
- Three hard-wired polynomials (CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

82.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 Ethernet (see note below)

NOTE

Although the CRC-32 Ethernet polynomial is used for CRC generation, the generation algorithm differs (MISR) from the one used by the Ethernet protocol (IEEE 802).

82.3 Block diagram

The top level diagram of the CRC module is given in the following figure. Refer to the chip configuration chapter for number of contexts in this module.

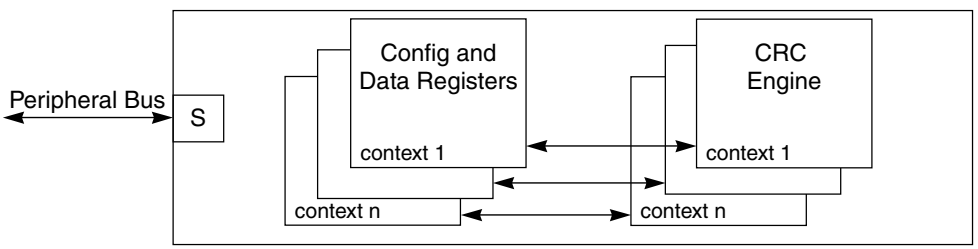


Figure 82-1. CRC top level diagram

82.4 Signal description

The CRC module does not generate any external signals.

82.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are read/write accessible in each access mode:

- user
- supervisor

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_ CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_ CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

82.5 CRC memory map and registers

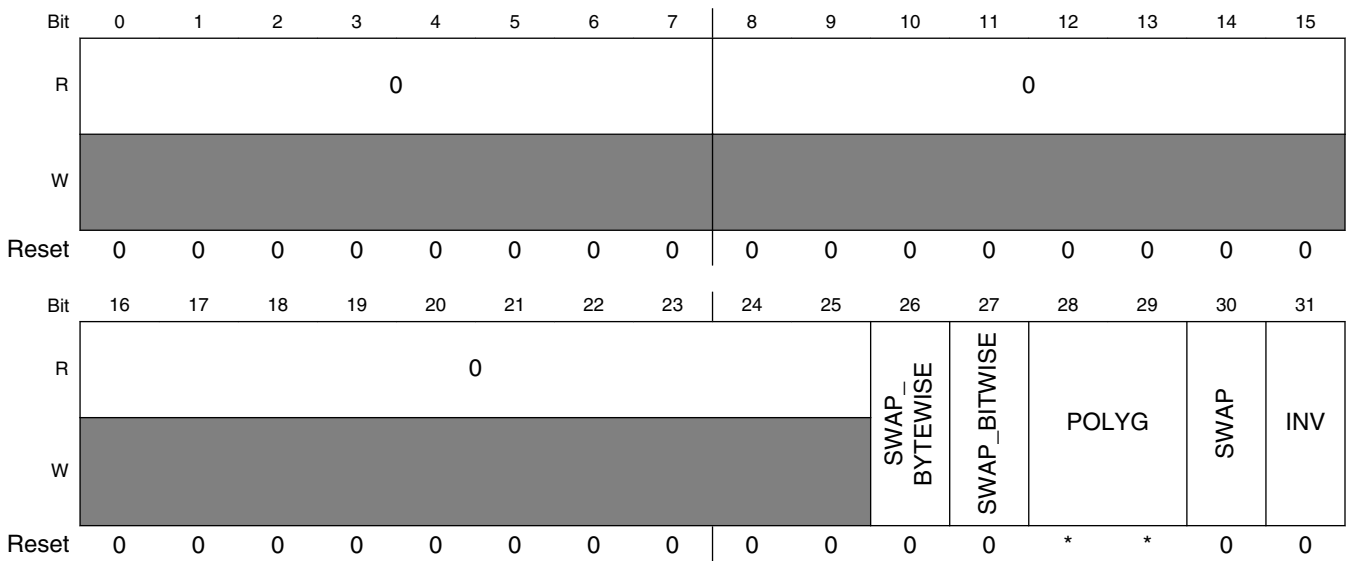
CRC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Configuration Register (CRC_CFG)	32	R/W	See section	82.5.1/4332
4	Input Register (CRC_INP)	32	R/W	0000_0000h	82.5.2/4333
8	Current Status Register (CRC_CSTAT)	32	R/W	FFFF_FFFFh	82.5.3/4334
C	Output Register (CRC_OUTP)	32	R	FFFF_FFFFh	82.5.4/4334

82.5.1 Configuration Register (CRC_CFG)

Access: User read/write.

Address: 0h base + 0h offset = 0h



* Notes:
- POLYG field: Reset value is a function of the number of contexts, N. When N%2 = 0, reset value = 01b. When N%2 = 1, reset value = 00b.

CRC_CFG field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWAP_ BYTEWISE	Swap CRC_INP byte-wise **NOTE:** INV and SWAP bits are set to 1 for CRC-32 polynomial calculations. 0 Do not swap 1 Perform byte-wise swap on CRC_INP input data internally for CRC-32 polynomial calculations.
27 SWAP_BITWISE	Swap CRC_INP bit-wise 0 Do not swap 1 Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.
28–29 POLYG	Polynomial selection

Table continues on the next page...

CRC_CFG field descriptions (continued)

Field	Description
	This bit can be written only during the configuration phase. 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 Reserved
30 SWAP	Swap selection This bit can be written only during the configuration phase. The swap operation is a bit-by-bit swapping of the content. 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content; in case of CRC-CCITT polynomial, the swap operation is applied on the 16 least-significant bits
31 INV	Inversion selection This bit can be written only during the configuration phase. The inversion operation is a complement (or negation) of the content. 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content In case of CRC-CCITT polynomial, the inversion operation is applied on the 16 least-significant bits.

82.5.2 Input Register (CRC_INP)

Access: User read/write.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC_INP field descriptions

Field	Description
0–31 INP	Input data for the CRC computation This register can be written with word-, halfword- (high or low), or byte-wide writes in any sequence. Only the bits written are fed to the CRC engine. In case of halfword write operation, the bytes must be contiguous.

82.5.3 Current Status Register (CRC_CSTAT)

Access: User read/write.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_CSTAT field descriptions

Field	Description
0–31 CSTAT	CRC signature status This register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 most-significant bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be written only during the configuration phase.

82.5.4 Output Register (CRC_OUTP)

Access: User read-only.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_OUTP field descriptions

Field	Description
0–31 OUTP	Final CRC signature This register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 MSB bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation.

82.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less or equal to the number of contexts.

The standard generator polynomials are given in Equation 81 on page 4335, Equation 82 on page 4335, and Equation 83 on page 4335 for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 81. CRC8 VDA CAN

$$x^{16} + x^{12} + x^5 + 1$$

Equation 82. CRC-CCITT (x.25 protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 83. CRC-32 (Ethernet protocol)

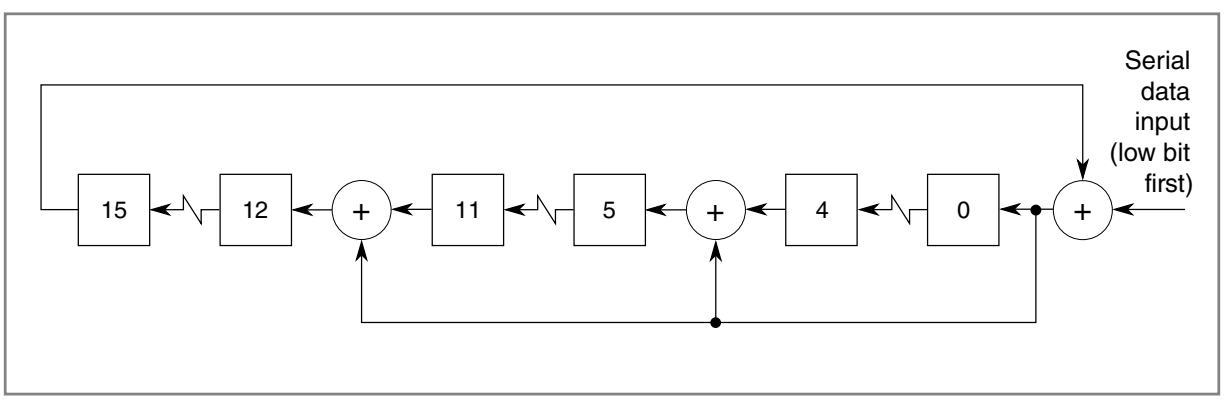


Figure 82-2. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. A diagram illustrating serial data loading of the CRC engine is shown in Figure 82-2 for the CRC-CCITT. Note that, conceptually, the least-significant, or "right-most," bit shown in Input Register (CRC_INP), is fed first into the engine. The actual

implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (high bit for low bit or low bit for high bit) and/or bit inversion of the final CRC signature can be applied to CRC_OUTP before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in Figure 82-3.

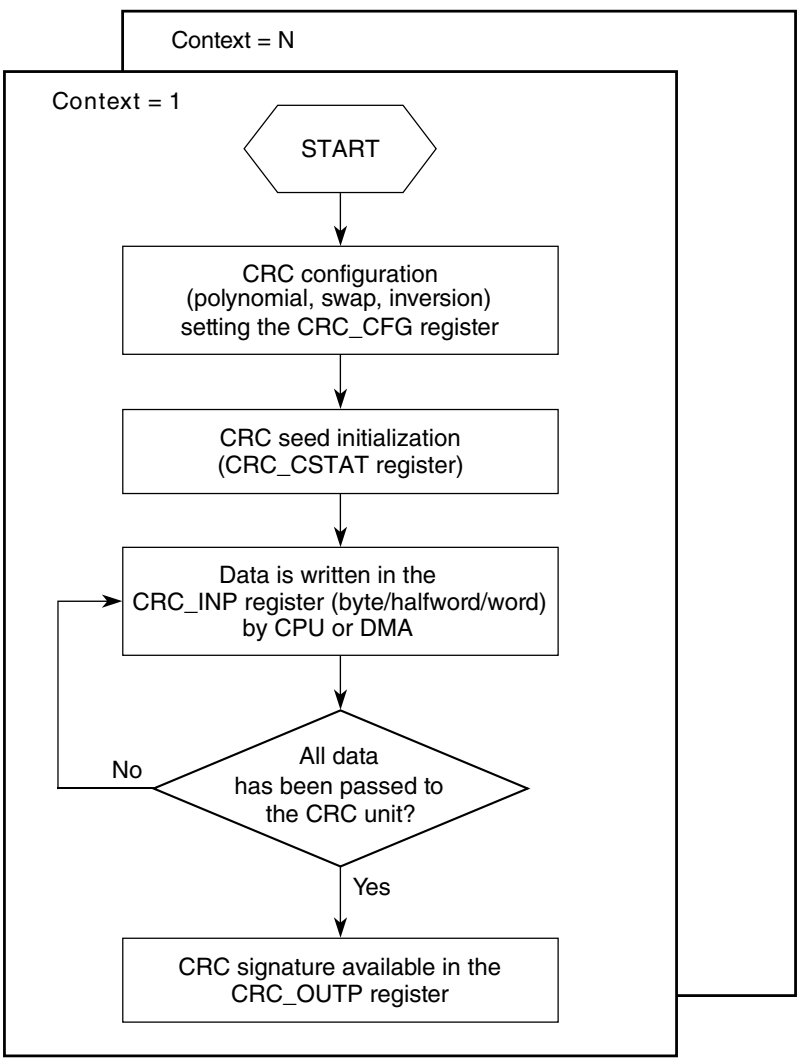


Figure 82-3. CRC computation flow

82.7 Use cases

82.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer—mem2mem, periph2mem, mem2periph—the following sequence, as shown in the following figure, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB-to-LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

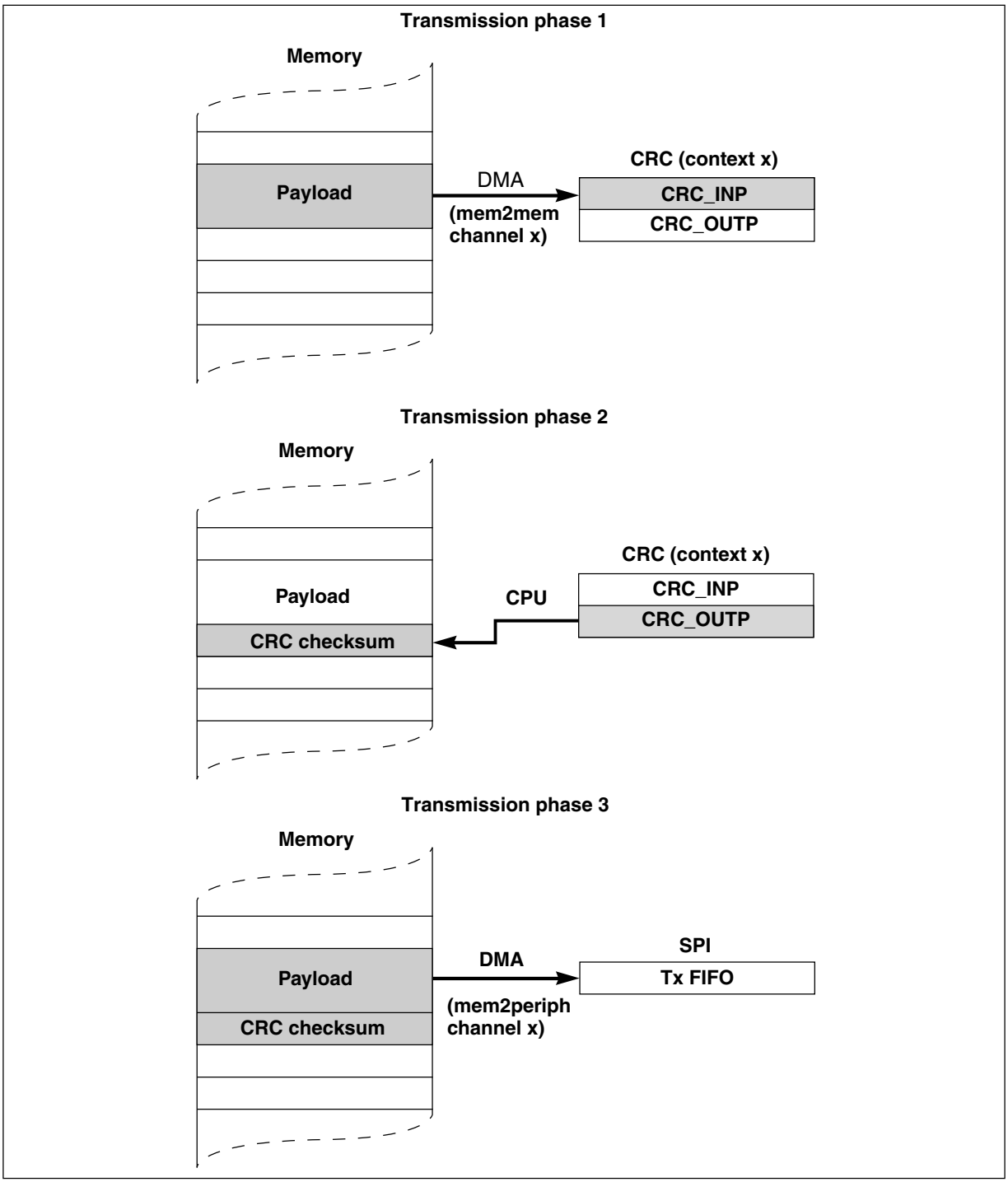


Figure 82-4. DMA-CRC transmission sequence

82.7.2 Register programming

- INV and/or SWAP (affecting the output only) can be applied to CRC-8, CRC-16, and CRC-32. Both can be applied together.

- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the Ethernet standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

Chapter 83
Memory Error Management Unit (MEMU)

83.1 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- System RAM
- Peripheral RAM
- Flash memory

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single Bit error in the data part that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Multiple bit error that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see the chip-specific MEMU information.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC and MBIST
- Peripheral RAM ECC
- Flash memory ECC

Each instance has two reporting tables. Correctable errors are reported in one table, and the uncorrectable errors are reported in the other.

See Design overview for the reporting block details.

83.2 Features

The MEMU has the following features:

- Supports up to 128 error sources per the following reporting blocks (the number of error sources are chip-specific; see the chip-specific MEMU information):
  - System RAM ECC and MBIST
  - Peripheral RAM ECC
  - Flash memory ECC
- Support for error reporting from the following category of error sources (both for ECC and MBIST):
  - System RAM
  - Peripheral RAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.

- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

83.3 Block diagram

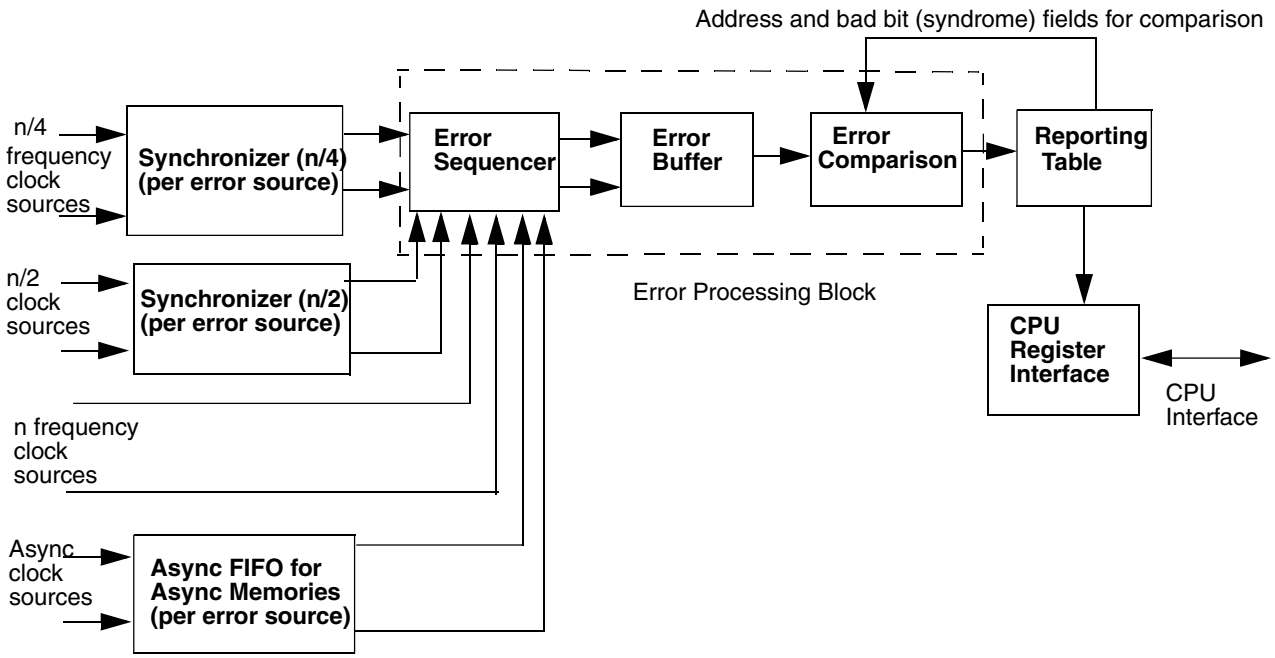


Figure 83-1. MEMU block diagram

83.4 Design overview

The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See Handling overflows (Multiple error reporting), for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

   For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

   This block is used to process the incoming errors coming from the device.

   - Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.
   - All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.
   - While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC or MBIST logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set., else the incoming error is dropped.

- While doing the error buffer uniqueness check, the type of error i.e Correctable Error (CE) or Uncorrectable Error UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (Depending on whether the incoming error is generated from ECC or MBIST logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique
- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

   This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

   The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

   The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (i.e. ..._CERR_STS or ..._UNCERR_STS). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

   The reporting table sizes are chip-specific; see the chip-specific MEMU information.

4. Register block

   The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

   The software, in principle, is not supposed to write VLD to 1 if it is already 1.

   The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

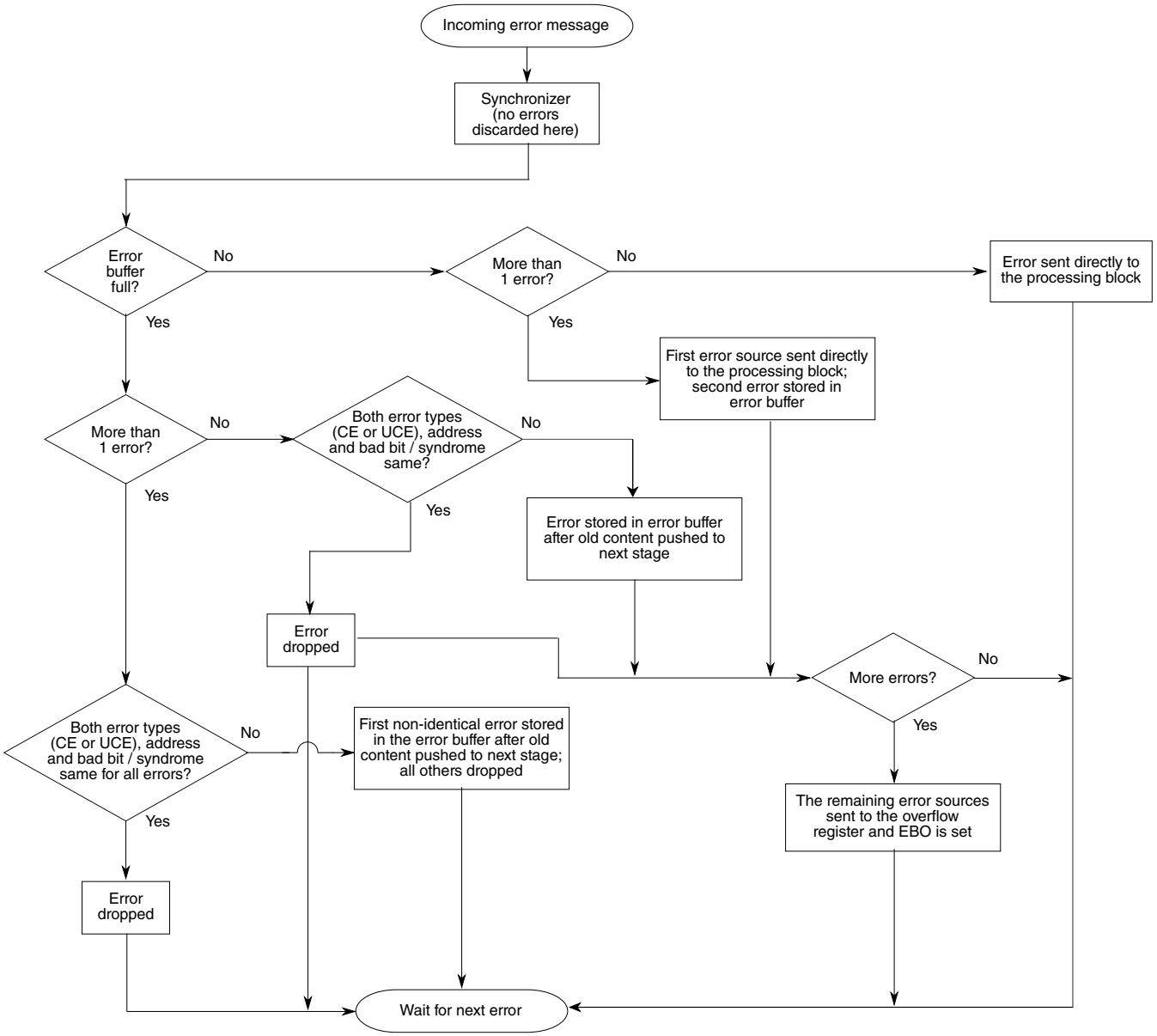


Figure 83-2. MEMU error buffer uniqueness check flow chart

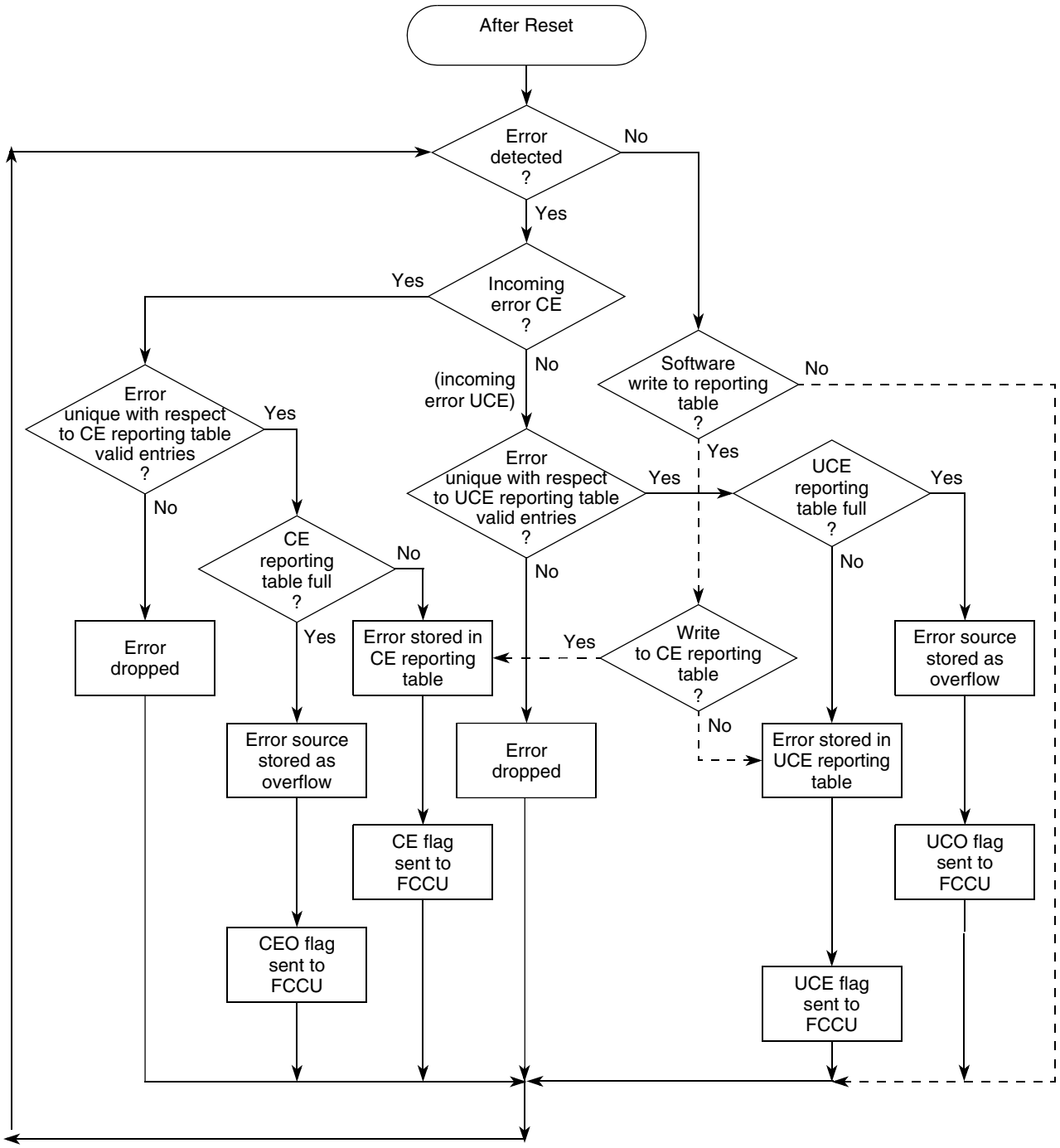


Figure 83-3. MEMU reporting table uniqueness check flow chart

83.5 External signal description

MEMU has no external signals.

83.6 Memory map and register definition

This section describes the registers on this module.

83.6.1 Overview

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

NOTE

The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

The Control, Error Flag and Debug registers are common for all MEMU error reporting blocks. The correctable error status, correctable error address, uncorrectable error status, uncorrectable error address and concurrent error overflow registers are replicated for each MEMU instance.

NOTE

The memory map in this section shows all the possible registers on this module and the associated offsets (which vary depending on the number of registers actually implemented). The actual availability, number and addresses of registers, and number of reported errors in the reporting table are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

MEMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (MEMU_CTRL)	32	R/W	0000_0000h	83.6.2/4352
4	Error flag register (MEMU_ERR_FLAG)	32	w1c	0000_0000h	83.6.3/4353
C	Debug register (MEMU_DEBUG)	32	R/W	0000_0000h	83.6.4/4356
20	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS0)	32	R/W	0000_0000h	83.6.5/4358
24	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR0)	32	R/W	0000_0000h	83.6.6/4359

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS1)	32	R/W	0000_0000h	83.6.5/4358
2C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR1)	32	R/W	0000_0000h	83.6.6/4359
30	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS2)	32	R/W	0000_0000h	83.6.5/4358
34	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR2)	32	R/W	0000_0000h	83.6.6/4359
38	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS3)	32	R/W	0000_0000h	83.6.5/4358
3C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR3)	32	R/W	0000_0000h	83.6.6/4359
40	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS4)	32	R/W	0000_0000h	83.6.5/4358
44	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR4)	32	R/W	0000_0000h	83.6.6/4359
48	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS5)	32	R/W	0000_0000h	83.6.5/4358
4C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR5)	32	R/W	0000_0000h	83.6.6/4359
50	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS6)	32	R/W	0000_0000h	83.6.5/4358
54	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR6)	32	R/W	0000_0000h	83.6.6/4359
58	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS7)	32	R/W	0000_0000h	83.6.5/4358
5C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR7)	32	R/W	0000_0000h	83.6.6/4359
60	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS8)	32	R/W	0000_0000h	83.6.5/4358
64	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR8)	32	R/W	0000_0000h	83.6.6/4359
68	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS9)	32	R/W	0000_0000h	83.6.5/4358
6C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR9)	32	R/W	0000_0000h	83.6.6/4359
70	System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)	32	R/W	0000_0000h	83.6.7/4359
74	System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	83.6.8/4360
78	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW0)	32	w1c	0000_0000h	83.6.9/4361
7C	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW1)	32	w1c	0000_0000h	83.6.9/4361

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW2)	32	w1c	0000_0000h	83.6.9/4361
620	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS0)	32	R/W	0000_0000h	83.6.10/ 4361
624	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR0)	32	R/W	0000_0000h	83.6.11/ 4362
628	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS1)	32	R/W	0000_0000h	83.6.10/ 4361
62C	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR1)	32	R/W	0000_0000h	83.6.11/ 4362
630	Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)	32	R/W	0000_0000h	83.6.12/ 4362
634	Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	83.6.13/ 4363
638	Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW0)	32	w1c	0000_0000h	83.6.14/ 4363
63C	Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW1)	32	w1c	0000_0000h	83.6.14/ 4363
C20	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS0)	32	R/W	0000_0000h	83.6.15/ 4364
C24	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR0)	32	R/W	0000_0000h	83.6.16/ 4365
C28	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS1)	32	R/W	0000_0000h	83.6.15/ 4364
C2C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR1)	32	R/W	0000_0000h	83.6.16/ 4365
C30	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS2)	32	R/W	0000_0000h	83.6.15/ 4364
C34	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR2)	32	R/W	0000_0000h	83.6.16/ 4365
C38	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS3)	32	R/W	0000_0000h	83.6.15/ 4364
C3C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR3)	32	R/W	0000_0000h	83.6.16/ 4365
C40	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS4)	32	R/W	0000_0000h	83.6.15/ 4364
C44	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR4)	32	R/W	0000_0000h	83.6.16/ 4365
C48	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS5)	32	R/W	0000_0000h	83.6.15/ 4364
C4C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR5)	32	R/W	0000_0000h	83.6.16/ 4365
C50	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS6)	32	R/W	0000_0000h	83.6.15/ 4364

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C54	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR6)	32	R/W	0000_0000h	83.6.16/ 4365
C58	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS7)	32	R/W	0000_0000h	83.6.15/ 4364
C5C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR7)	32	R/W	0000_0000h	83.6.16/ 4365
C60	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS8)	32	R/W	0000_0000h	83.6.15/ 4364
C64	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR8)	32	R/W	0000_0000h	83.6.16/ 4365
C68	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS9)	32	R/W	0000_0000h	83.6.15/ 4364
C6C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR9)	32	R/W	0000_0000h	83.6.16/ 4365
C70	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS10)	32	R/W	0000_0000h	83.6.15/ 4364
C74	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR10)	32	R/W	0000_0000h	83.6.16/ 4365
C78	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS11)	32	R/W	0000_0000h	83.6.15/ 4364
C7C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR11)	32	R/W	0000_0000h	83.6.16/ 4365
C80	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS12)	32	R/W	0000_0000h	83.6.15/ 4364
C84	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR12)	32	R/W	0000_0000h	83.6.16/ 4365
C88	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS13)	32	R/W	0000_0000h	83.6.15/ 4364
C8C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR13)	32	R/W	0000_0000h	83.6.16/ 4365
C90	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS14)	32	R/W	0000_0000h	83.6.15/ 4364
C94	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR14)	32	R/W	0000_0000h	83.6.16/ 4365
C98	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS15)	32	R/W	0000_0000h	83.6.15/ 4364
C9C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR15)	32	R/W	0000_0000h	83.6.16/ 4365
CA0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS16)	32	R/W	0000_0000h	83.6.15/ 4364
CA4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR16)	32	R/W	0000_0000h	83.6.16/ 4365
CA8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS17)	32	R/W	0000_0000h	83.6.15/ 4364

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CAC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR17)	32	R/W	0000_0000h	83.6.16/ 4365
CB0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS18)	32	R/W	0000_0000h	83.6.15/ 4364
CB4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR18)	32	R/W	0000_0000h	83.6.16/ 4365
CB8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS19)	32	R/W	0000_0000h	83.6.15/ 4364
CBC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR19)	32	R/W	0000_0000h	83.6.16/ 4365
CC0	Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)	32	R/W	0000_0000h	83.6.17/ 4365
CC4	Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)	32	R/W	0000_0000h	83.6.18/ 4366
CC8	Flash memory concurrent overflow register (MEMU_FLASH_OFLW0)	32	w1c	0000_0000h	83.6.19/ 4366

83.6.2 Control register (MEMU_CTRL)

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWR	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_CTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.

Table continues on the next page...

MEMU_CTRL field descriptions (continued)

Field	Description
17–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

83.6.3 Error flag register (MEMU_ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Access: User read/write

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0			SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_ERR_FLAG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: • a new Flash ECC uncorrectable error was detected. • a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SR_CE	System RAM ECCand MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECCand MBISTcorrectable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECCand MBIST Uncorrectable Error Detect flag. Asserted when: • A new System RAM ECCand MBIST uncorrectable error is detected • A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
29 SR_CEO	System RAM ECCand MBIST Correctable error Overflow flag. Asserted when the System RAM ECCand MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECCand MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECCand MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECCand MBIST Error buffer Overflow. Flag asserted when the internal error processing buffer of System RAM ECCand MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

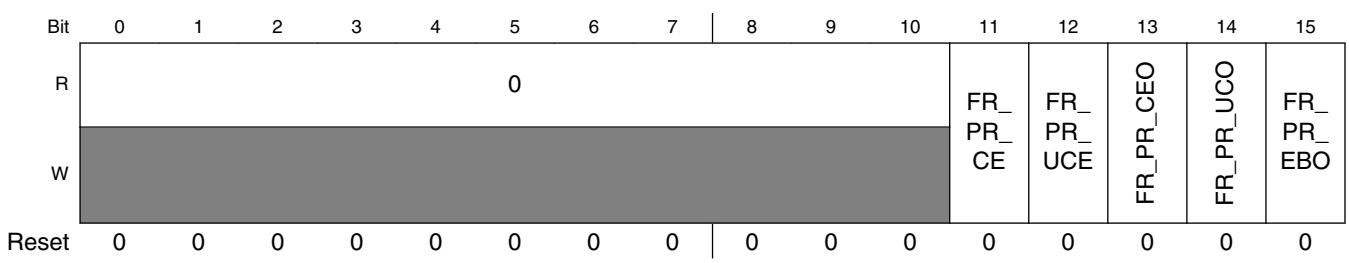
83.6.4 Debug register (MEMU_DEBUG)

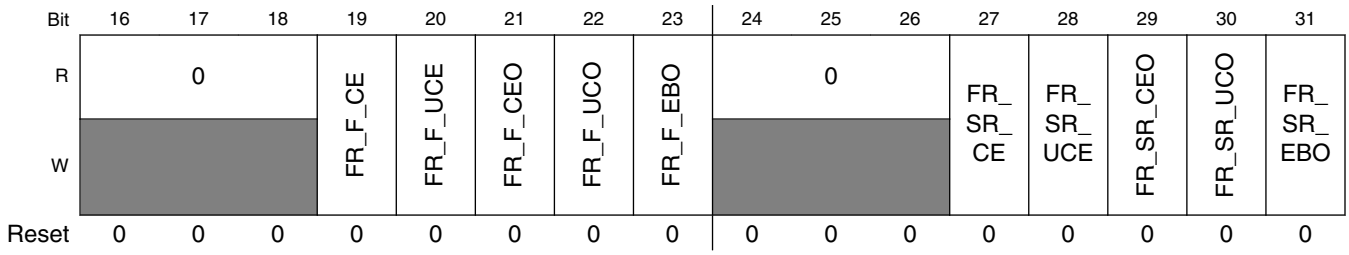
This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Access: User read/write

Address: 0h base + Ch offset = Ch





MEMU_DEBUG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

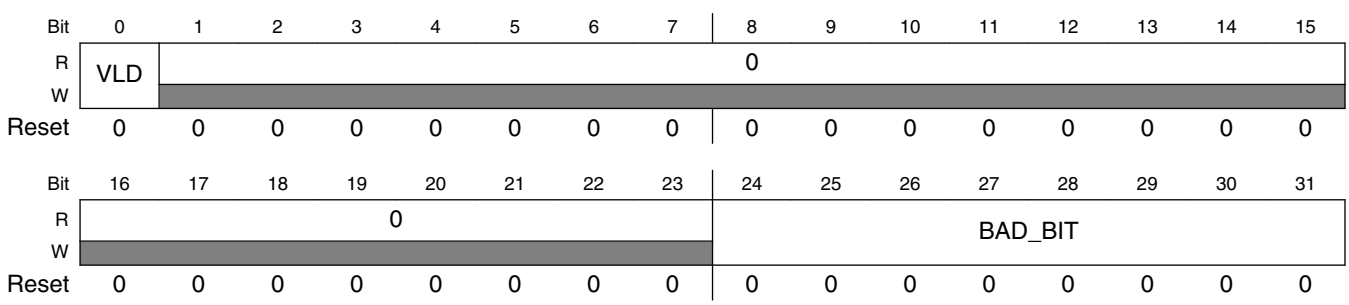
83.6.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 20h offset + (8d × i), where i=0d to 9d



MEMU_SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

83.6.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 24h offset + (8d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

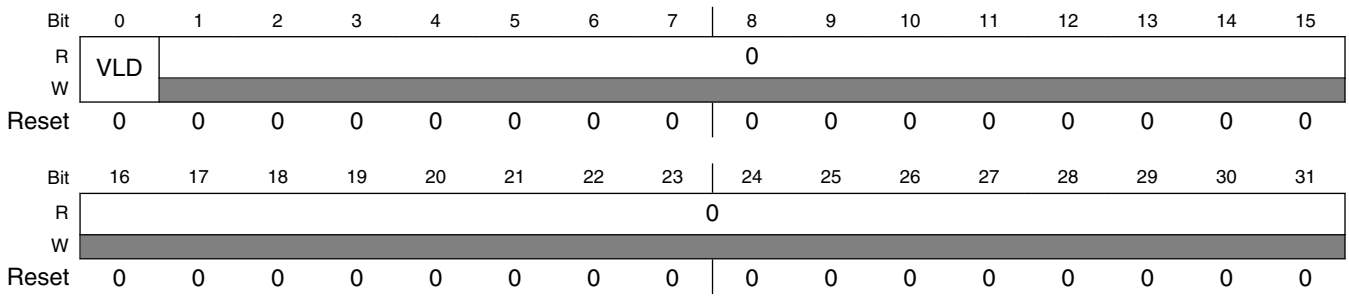
83.6.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 70h offset = 70h



MEMU_SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

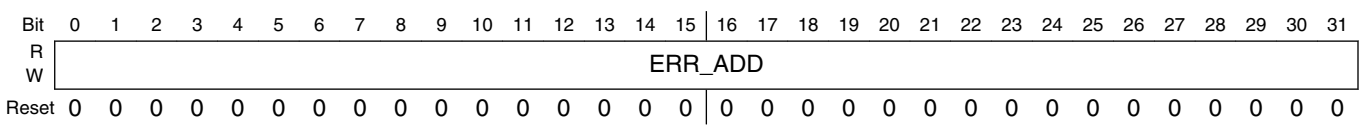
83.6.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 74h offset = 74h



MEMU_SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (SYS_RAM_UNCERR_STS[VLD]) = 0.

83.6.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 78h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

83.6.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 620h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted.

Table continues on the next page...

MEMU_PERIPH_RAM_CERR_STS*n* field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

83.6.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 624h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

83.6.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)

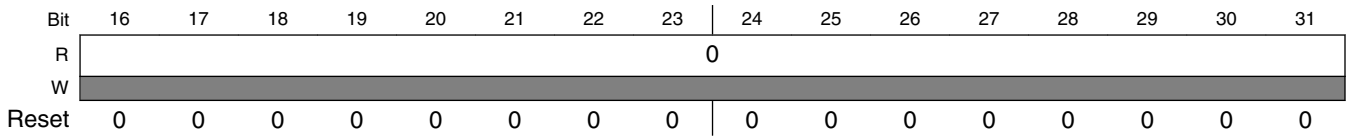
Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 630h offset = 630h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MEMU_PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

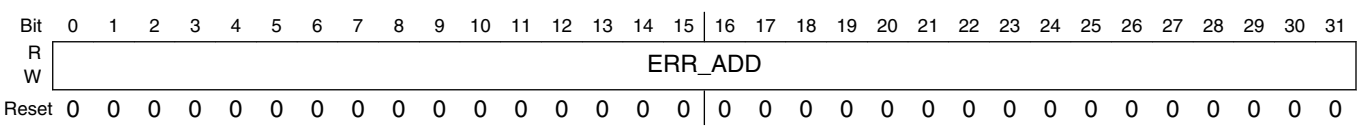
83.6.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 634h offset = 634h



MEMU_PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (PERIPH_RAM_UNCERR_STS[VLD]) = 0.

83.6.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 638h offset + (4d × i), where i=0d to 1d

Bit	0–15	16–31
R	OFLW	
W	w1c	
Reset	0	0

MEMU_PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

83.6.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + C20h offset + (8d × i), where i=0d to 19d

Bit	0	1–7	8–15
R	VLD	0	
W			
Reset	0	0	0

Bit	16–23	24–31
R	0	BAD_BIT
W		
Reset	0	0

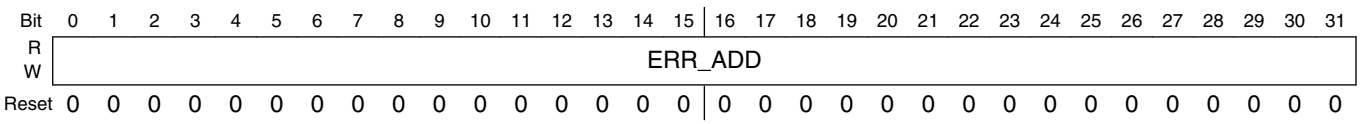
MEMU_FLASH_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid . Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in flash memory where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

83.6.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + C24h offset + (8d × i), where i=0d to 19d



MEMU_FLASH_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

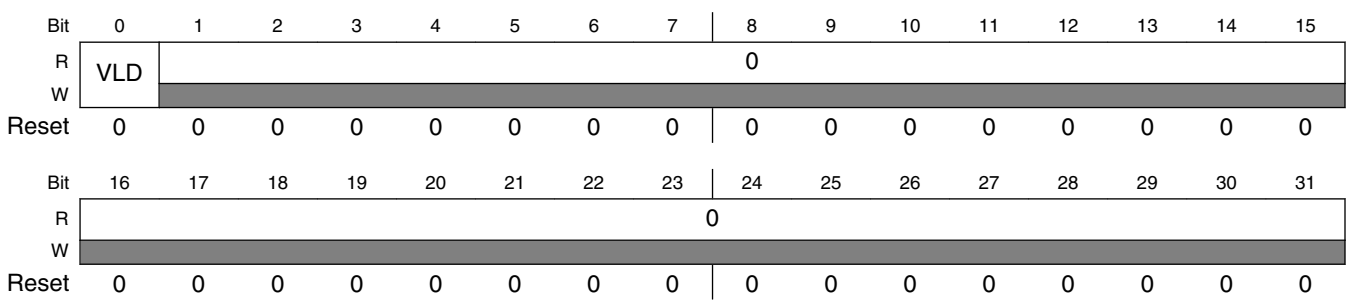
83.6.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC0h offset = CC0h



MEMU_FLASH_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_FLASH_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

83.6.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC4h offset = CC4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (FLASH_UNCERR_STS[VLD]) = 0.

83.6.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC8h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

83.7 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

83.7.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit ='1' for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address. The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (i.e. duplicate entries).

83.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

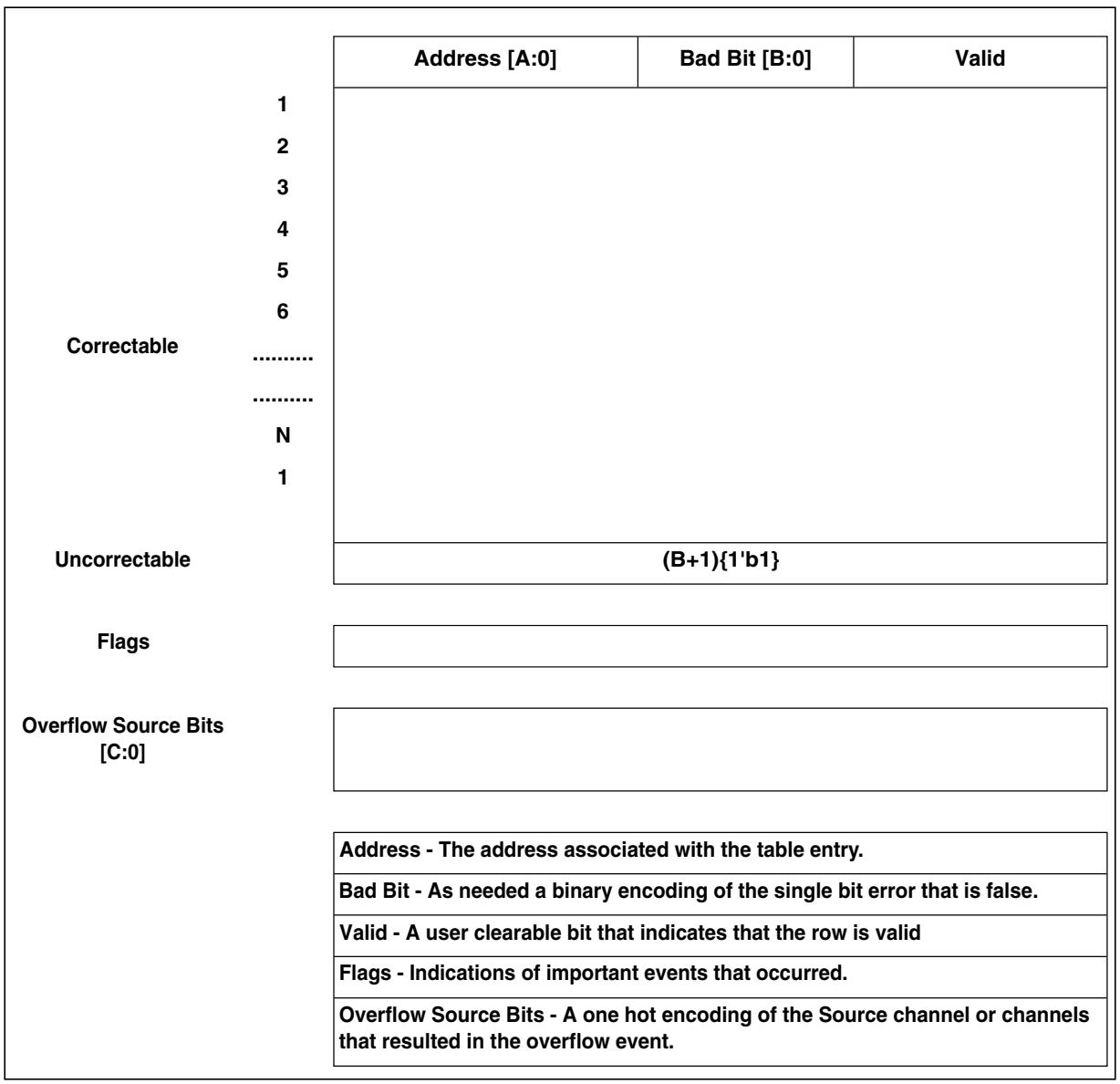


Figure 83-4. Generic representation of the reporting table of one MEMU reporting block

83.7.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
    - More than 2 errors detected at same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not "guarantee" that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together with two (or more) additional errors at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Error Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
    - An ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
3. The Overflows will be stored in the bit-wise order i.e., if the error bit 31 reports an error, the bit 31 of the overflow register will be set to '1'.

Chapter 84
Indirect Memory Access (IMA)

84.1 Introduction

Indirect Memory Access (IMA) refers to the activity of accessing any chip memory for the purpose of reading and/or modifying data, including ECC check bits. This capability is useful for test activities, e.g., verifying ECC functionality.

The MPC5777M MCU provides two mechanisms for IMA:

- The built-in CPU End2End ECC test capability provides a path for accessing data and ECC check bits from some SRAM arrays
- An IMA controller module provides registers for selecting, reading, and writing memory data.

Note

Not all SRAM data and ECC bits are accessible. See the Device Configuration chapter for details.

84.2 Accessing memory via CPU End2End ECC test

The processor cores can access all bits (including ECC check bits) in SRAM arrays that support E2E ECC with the exceptions of addresses that are blocked by MPU configuration.

Note

All memory accesses by the processor cores are filtered through the MPU. There are no back doors to blocked memory addresses.

SRAM is selected by absolute memory address with check bit data in a special register. The register used depends on which memory is being accessed—I-cache/D-cache, or other memory.

Since all CPU memory accesses are arbitrated by the crossbar (XBAR), there is no need to halt normal memory access in software.

Memory is accessed via processor core registers. The register(s) used depends on the memory being accessed. This is described in the following sections.

84.2.1 Accessing I-cache and D-cache memory via processor core

For I-cache and D-cache memory, the SRAM location is selected by cache way, set and word with data and check bits in registers provided specifically for reading and writing cache tag bits and ECC check bits. There are two Device Control Registers (DCRs) provided for cache access:

- Cache Debug Access Control Register (CDACNTL)
- Cache Debug Access Data Register (CDADATA)

These registers are summarized in the following sections.

84.2.1.1 Cache Debug Access Control register (CDACNTL)

The Cache Debug Access Control Register (CDACNTL) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

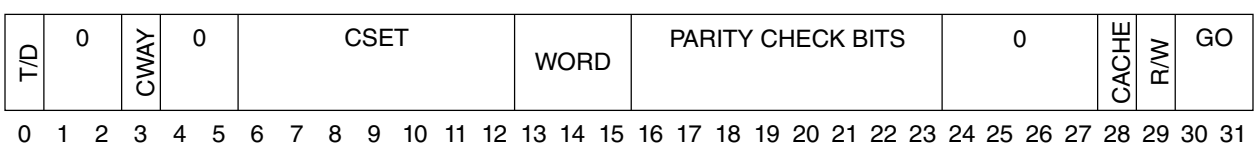


Figure 84-1. Cache Debug Access Control Register (CDACNTL)

The CDACNTL bits are described in Table 84-1.

Table 84-1. CDACNTL field descriptions

Bits	Name	Description
0	T/D	Tag / Data: 0 Data array selected

Table continues on the next page...

Table 84-1. CDACNTL field descriptions (continued)

Bits	Name	Description
		1 Tag array selected
1–2	—	Reserved - read as zeros
3	CWAY	Cache Way. Specifies the cache way to be selected
4–5	—	Reserved - read as zeros
6–12	CSET	Cache Set. Specifies the cache set to be selected. Only valid set numbers for the selected cache are supported, otherwise results are undefined.
13–15	WORD	Word (Used for data array access only, I or D-cache) Specifies one of eight words of selected set
16–23	PARITY CHECK BITS	Parity check bits (I or D-cache) D-cache Data array, I-cache Data Array: parity check bits for data. Tag Array: parity check bits for tag.
24–27	—	Reserved - read as zeros
28	CACHE	Cache Select. Specifies the cache to be selected. 0 Selects the data cache for the operation. 1 Selects the instruction cache for the operation.
29	R/W	Read / Write: 0 Selects write operation. Write the data in the CDADATA register to the location specified by this CDACNTL register. 1 Selects read operation. Read the cache memory location specified by this CDACNTL register and store the resulting data in the CDADATA register and store the parity bits in this CDACNTL register.
30–31	GO	GO command bits 00 Inactive or complete (no action taken) hardware sets GO=00 when an operation is complete 01 Read or write cache memory location specified by this CDACNTL register. 1x Reserved

84.2.1.2 Cache Debug Access Data Register (CDADATA)

The Cache Debug Access Data Register (CDADATA) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

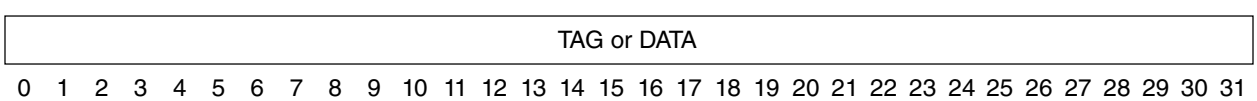


Figure 84-2. Cache Debug Access Data Register (CDADATA)

The CDADATA bits are described in Table 84-2.

Table 84-2. CDADATA field descriptions

Bits	Name	Description
0–31	TAG	TAG Array Access Data - when accessing the tag array of the I-cache: 0–19 Tag compare bits 20–22 Reserved 23 Valid bit 24 R bit 25 SO bit 26–28 Reserved 29–31 Lockout bits. These three bits should have the same value, 1-Locked out, 0- not locked out. TAG Array Access Data - when accessing the tag array of the D-cache: 0–20 Tag compare bits 21–22 Reserved 23 Valid bit 24 R bit 25 SO bit 26 SW bit 27 UW bit 28 Reserved 29–31 Lockout bits. These three bits should have the same value, 1-Locked out, 0-notlocked out.
	DATA	DATA Array Access Data (Bytes 0:3 of the selected word) - when accessing the data array of either cache: 0–7 byte 0 8–15 byte 1 16–23 byte 2 24–31 byte 3

84.2.2 Accessing non I-cache or D-cache memory via processor core

The e2eECC Error Control/Status Register 0 (E2EECSR0) is provided specifically for reading and writing E2E ECC check bits. This register is described in the following section.

84.2.2.1 e2eECC Error Control/Status Register 0 (E2EECSR0)

The e2eECC Error Control/Status Register 0 (E2EECSR0) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to E2E ECC data. It is shown in the following figure.

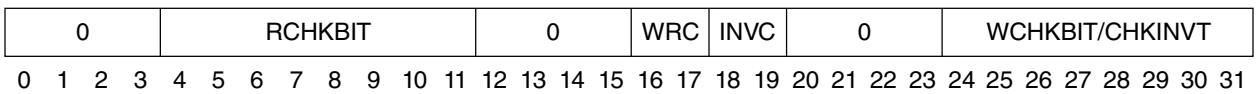


Figure 84-3. e2eECC Error Control/Status Register 0 (E2EECSR0)

The E2EECSR0 bits are described in Table 84-3.

Table 84-3. CDACNTL field descriptions

Bits	Name	Description
0–3	—	Reserved - read as zeros
4–11	RCHKBIT	Read Checkbits. This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the data BIU. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation, assuming interrupts are masked.
12–15	—	Reserved - read as zeros
16–17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10, by driving the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved **Note:** Checkbit substitution has priority over Error injection
18–19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying p_hwchkbit[7:0] based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying p_hwchkbit[7:0] based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved **Note:** Checkbit substitution has priority over Error injection
20–23	—	Reserved - read as zeros

Table continues on the next page...

Table 84-3. CDACNTL field descriptions (continued)

Bits	Name	Description
24–31	WCHKBIT/CHKINVT	Write Checkbits / Checkbit Invert Mask. This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out when error injection occurs.

84.3 Accessing memory via the IMA module

The Indirect Memory Access (IMA) module provides an alternative way to access on-chip memory without going through standard peripheral or system RAM interfaces to check the data or change the data.

This alternative path can be used for diagnostic purposes. For example, for SRAM arrays that do not support E2E ECC, the IMA block provides a way to write corrupted ECC values so that the ECC error decoding logic can be checked.

Please note the following restrictions on memory access via IMA:

- ECC must be disabled prior to IMA write access. If ECC is enabled and a write access is attempted to ECC-bits, the write access is blocked and an error signal is forwarded to the FCCU and the corresponding error flag is set. Read access is always allowed.
- The IMA must be unlocked for reads

The order of precedence for access to a memory is shown below. Once the IMA has been granted access, no other accesses are allowed until the IMA select line to the selected memory is negated by the IMA.

Note

Once the IMA has selected a RAM, it expects no other accesses from any other master until deselected by the IMA.

The IMA module signals the FCCU when it is reading or writing to a memory address. This enables the FCCU to assess whether the access is intentional or spurious. For an intentional access the respective FCCU error input needs to be disabled, otherwise it should be enabled.

84.3.1 Features

The IMA module includes the following features:

- Able to read all bits in all RAM arrays connected to the IMA, once the IMA is unlocked for reads
- Enables reading and manipulation of ECC check bits.
- During an IMA SRAM access, other accesses are affected as follows:
  - Reads return random data.
  - Writes have no effect.
- Any IMA access to memory are indicated in the Fault Collection and Control Unit (FCCU).
- IMA accesses to memory do not bypass any SRAM repair which may have occurred during production test of the device.
- IMA accesses to memory use a physical address, not a logical address, and thus memory addresses used by the IMA are not memory mapped.

84.3.2 Block diagram

The following figure shows the structure and function of the IMA module. The Register Control Block contains the memory-mapped registers accessible by software. The Safety Block prevents spurious activation of the IMA module. It ensures the IMA does not access RAM unless the system needs it to.

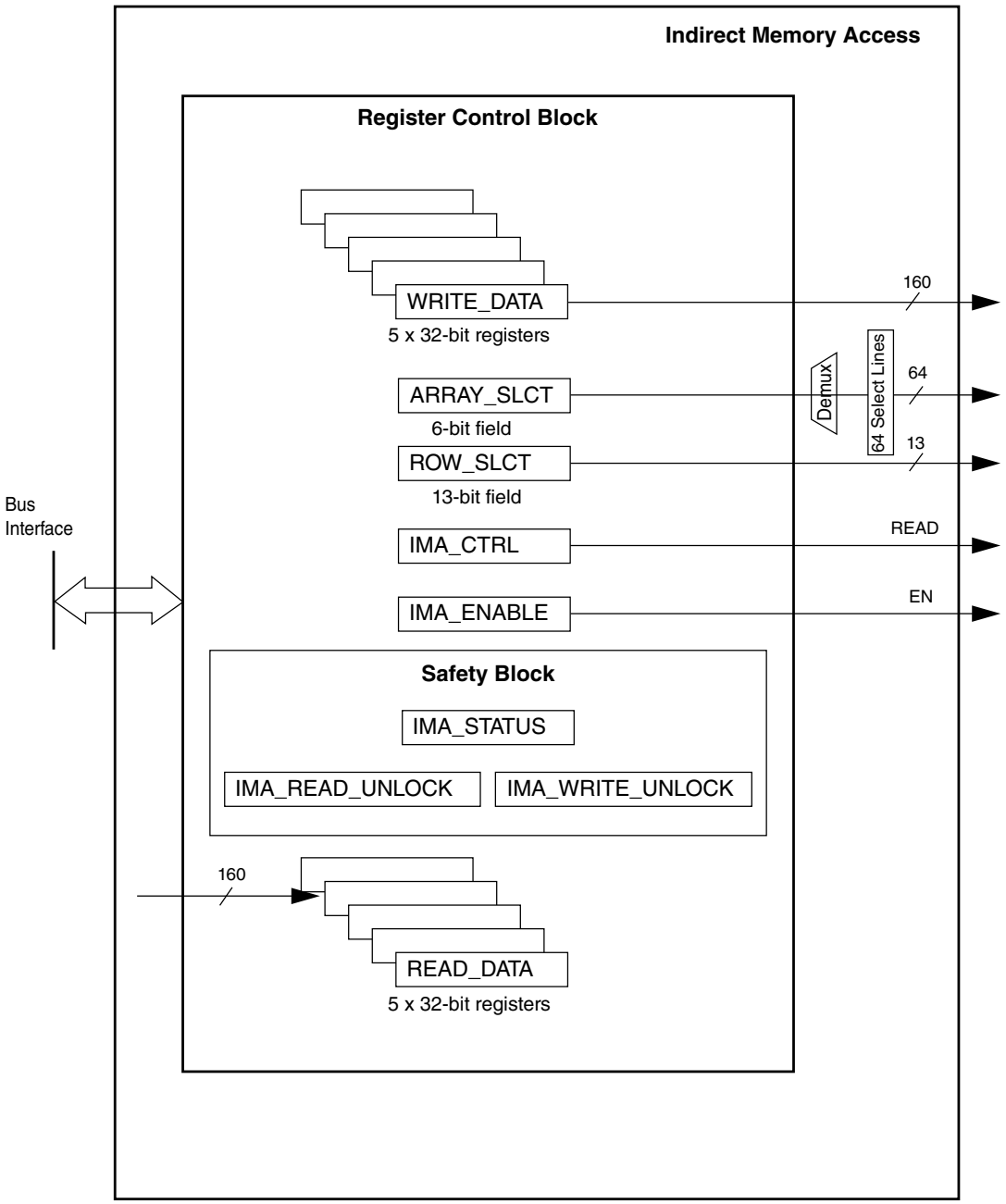


Figure 84-4. IMA block diagram

84.3.3 IMA Module Memory Map and Registers

This section provides details of the IMA memory map. It also provides detailed descriptions for each of the registers in that map .

The Indirect Memory Access unit's memory map allows for the CPU to access RAM's on the chip to write test data so that single-bit ECCs and multi-bit ECC logic can be tested.

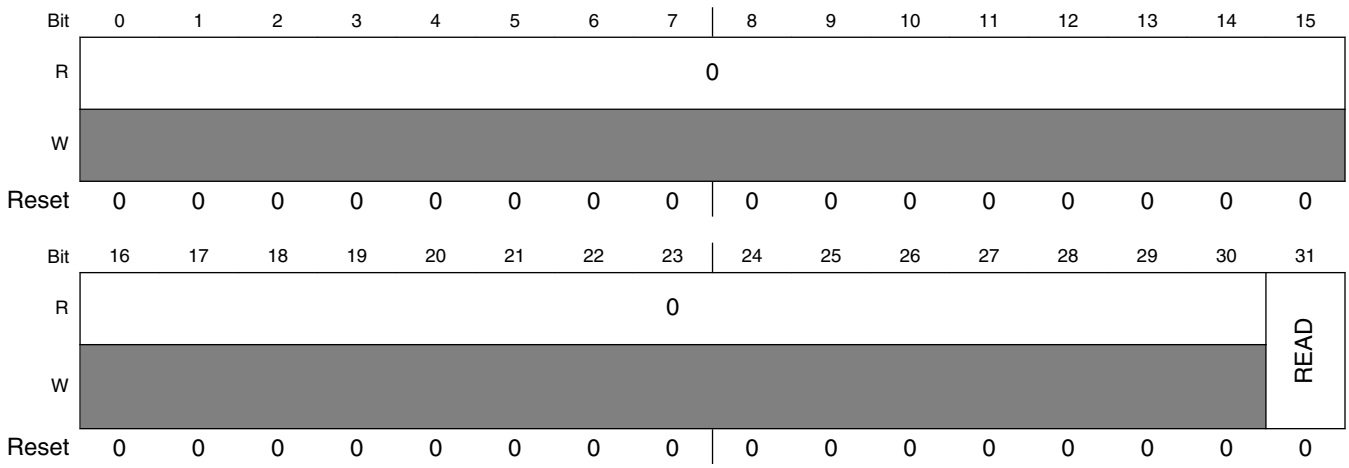
IMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (IMA_CTRL)	32	R/W	0000_0000h	84.3.3.1/ 4380
4	Enable Access Register (IMA_ENABLE)	32	W	0000_0000h	84.3.3.2/ 4381
8	Status Register (IMA_STATUS)	32	R	0000_0101h	84.3.3.3/ 4382
C	RAM Select Register (IMA_SLCT)	32	R/W	0000_0000h	84.3.3.4/ 4383
10	Write Unlock Register (IMA_WRITE_UNLOCK)	32	R/W	0000_0000h	84.3.3.5/ 4384
14	Read Unlock Register (IMA_READ_UNLOCK)	32	R/W	0000_0000h	84.3.3.6/ 4384
2C	RAM Write Data Register 4 (IMA_WRITE_DATA_4)	32	R/W	0000_0000h	84.3.3.7/ 4385
30	RAM Write Data Register 3 (IMA_WRITE_DATA_3)	32	R/W	0000_0000h	84.3.3.8/ 4385
34	RAM Write Data Register 2 (IMA_WRITE_DATA_2)	32	R/W	0000_0000h	84.3.3.9/ 4386
38	RAM Write Data Register 1 (IMA_WRITE_DATA_1)	32	R/W	0000_0000h	84.3.3.10/ 4386
3C	RAM Write Data Register 0 (IMA_WRITE_DATA_0)	32	R/W	0000_0000h	84.3.3.11/ 4387
4C	RAM Read Data Register 4 (IMA_READ_DATA_4)	32	R	0000_0000h	84.3.3.12/ 4387
50	RAM Read Data Register 3 (IMA_READ_DATA_3)	32	R	0000_0000h	84.3.3.13/ 4388
54	RAM Read Data Register 2 (IMA_READ_DATA_2)	32	R	0000_0000h	84.3.3.14/ 4389
58	RAM Read Data Register 1 (IMA_READ_DATA_1)	32	R	0000_0000h	84.3.3.15/ 4389
5C	RAM Read Data Register 0 (IMA_READ_DATA_0)	32	R	0000_0000h	84.3.3.16/ 4390

84.3.3.1 Control Register (IMA_CTRL)

The IMA CTRL register controls read/write accesses. Since the IMA may interfere with proper read/write access of RAMs, a locking scheme is implemented to prevent accidental activation of the IMA. Using this scheme, the IMA read/write accesses can be locked when desired.

Address: 0h base + 0h offset = 0h



IMA_CTRL field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 READ	IMA Read. This bit controls whether the IMA will perform a read or write 0 The IMA access will be a write. 1 The IMA access will be a read.

84.3.3.2 Enable Access Register (IMA_ENABLE)

The Enable Access Register is a single-bit register that starts an IMA access. The bit automatically clears itself after two clocks thus providing an enable pulse.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															0
W																EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

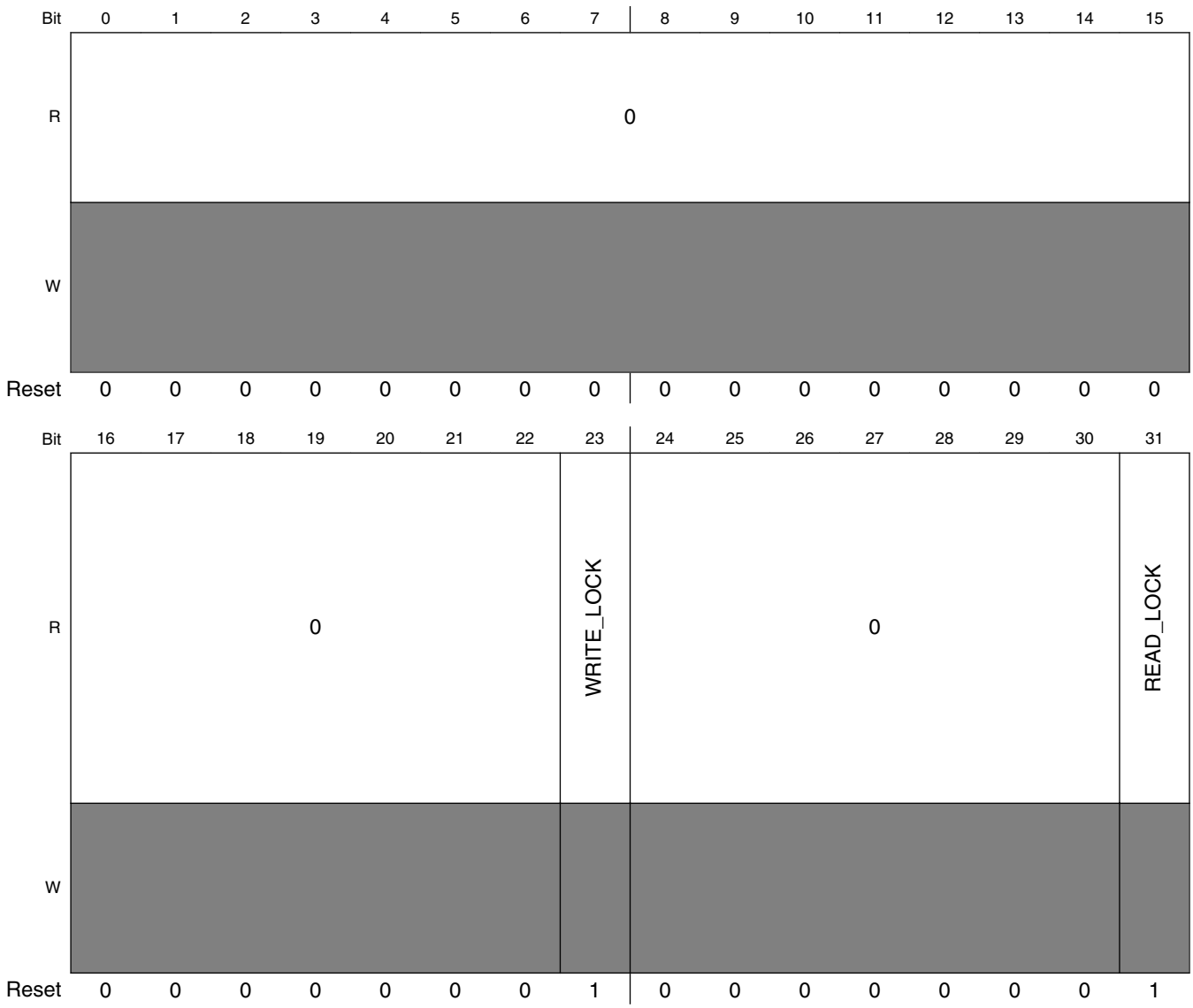
IMA_ENABLE field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 EN	IMA Enable. This bit controls RAM access initiation by the IMA. Once set, this bit is cleared after two clocks to provide an enable pulse. 0 No IMA access will occur or the IMA has finished the previous access. 1 Start an IMA access or an IMA access is occurring.

84.3.3.3 Status Register (IMA_STATUS)

The IMA Status Register shows the current status of the IMA as to which operations are unlocked.

Address: 0h base + 8h offset = 8h



IMA_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

IMA_STATUS field descriptions (continued)

Field	Description
23 WRITE_LOCK	The ability of the IMA to do write accesses. The WRITE_LOCK bit shows the current status of the lock on IMA write accesses. If set, the IMA does not write to the memories. If cleared, the IMA is allowed to do writes to the memories. 0 Write accesses from the IMA to memories are allowed 1 Write accesses from the IMA to memories are not allowed. Any memory writes from the IMA are ignored.
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 READ_LOCK	The ability of the IMA to do read accesses. The READ_LOCK bit shows the current status of the lock on IMA read accesses. If set, the IMA does not do reads from the memories. If cleared, the IMA is allowed to do reads from the memories. 0 Read accesses from the IMA to memories are allowed. 1 Read accesses from the IMA to memories are not allowed. Any memory reads from the IMA are ignored.

84.3.3.4 RAM Select Register (IMA_SLCT)

The IMA RAM Select register provides a method of selecting which RAM memory block will be accessed. This method is used because RAM is to be accessed in a non-functional way, i.e., accessing all bits including the ECC, and accessing on RAM word boundaries instead of byte or 32-bit word boundary.

NOTE

Only one RAM can be selected at a time. Each SoC has a different encoding for the RAMs. See the Device Configuration chapter for details.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			ROW_SLCT													0										ARRAY_SLCT					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_SLCT field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–15 ROW_SLCT	The ROW_SLCT value specifies the row of the current RAM array under test to be selected. Hence the largest RAM which the IMA can access is an 8 KB word RAM.
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

IMA_SLCT field descriptions (continued)

Field	Description
26–31 ARRAY_SLCT	The ARRAY_SLCT specifies the RAM array to be tested. Up to 64 RAM instances can be supported.

84.3.3.5 Write Unlock Register (IMA_WRITE_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The WRITE_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for writes .

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE_KEY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_WRITE_UNLOCK field descriptions

Field	Description
0–31 WRITE_KEY	Key to unlock the IMA. • 0x04A43F95 is the first key that has to be written to start the unlock process. • 0xE4A9EBF7 is the second key that has to be written to unlock the IMA.

84.3.3.6 Read Unlock Register (IMA_READ_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The READ_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for reads .

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	READ_KEY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_READ_UNLOCK field descriptions

Field	Description
0–31 READ_KEY	Key to unlock the IMA Read Access to RAMs. • 0xF06AB5BC is the first key that has to be written to start the unlock process. • 0x14081B56 is the second key that has to be written to unlock the IMA.

84.3.3.7 RAM Write Data Register 4 (IMA_WRITE_DATA_4)

The IMA_WRITE_DATA_n registers hold the data values to be written to the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE_4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_WRITE_DATA_4 field descriptions

Field	Description
0–31 WRITE_4	Contains write data bits[159:128] to be written to RAM.

84.3.3.8 RAM Write Data Register 3 (IMA_WRITE_DATA_3)

The IMA_WRITE_DATA_n registers hold the data values to be written to the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE_3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_WRITE_DATA_3 field descriptions

Field	Description
0–31 WRITE_3	Contains write data bits[127:96] to be written to RAM.

84.3.3.9 RAM Write Data Register 2 (IMA_WRITE_DATA_2)

The IMA_WRITE_DATA_n registers hold the data values to be written to the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE_2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_WRITE_DATA_2 field descriptions

Field	Description
0–31 WRITE_2	Contains write data bits[95:64] to be written to RAM.

84.3.3.10 RAM Write Data Register 1 (IMA_WRITE_DATA_1)

The IMA_WRITE_DATA_n registers hold the data values to be written to the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_WRITE_DATA_1 field descriptions

Field	Description
0–31 WRITE_1	Contains write data bits[32:63] to be written to RAM.

84.3.3.11 RAM Write Data Register 0 (IMA_WRITE_DATA_0)

The IMA_WRITE_DATA_n registers hold the data values to be written to the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_WRITE_DATA_0 field descriptions

Field	Description
0–31 WRITE_0	Contains write data bits[31:0] to be written to RAM.

84.3.3.12 RAM Read Data Register 4 (IMA_READ_DATA_4)

The IMA_READ_DATA_n registers hold the data values read from the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_n registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	READ_4																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_READ_DATA_4 field descriptions

Field	Description
0–31 READ_4	Contains read data bits[159:128] to be written to RAM.

84.3.3.13 RAM Read Data Register 3 (IMA_READ_DATA_3)

The IMA_READ_DATA_n registers hold the data values read from the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_n registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	READ_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_READ_DATA_3 field descriptions

Field	Description
0–31 READ_3	Contains read data bits[127:96] to be written to RAM.

84.3.3.14 RAM Read Data Register 2 (IMA_READ_DATA_2)

The IMA_READ_DATA_n registers hold the data values read from the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_n registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	READ_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMA_READ_DATA_2 field descriptions

Field	Description
0–31 READ_2	Contains read data bits[95:64] to be written to RAM.

84.3.3.15 RAM Read Data Register 1 (IMA_READ_DATA_1)

The IMA_READ_DATA_n registers hold the data values read from the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_n registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	READ_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

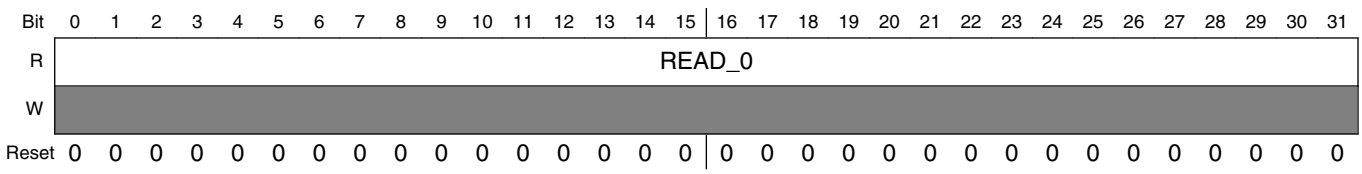
IMA_READ_DATA_1 field descriptions

Field	Description
0–31 READ_1	Contains read data bits[32:63] to be written to RAM.

84.3.3.16 RAM Read Data Register 0 (IMA_READ_DATA_0)

The IMA_READ_DATA_n registers hold the data values read from the RAM. The number of active data bits depends on the number of IOs for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_n registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Address: 0h base + 5Ch offset = 5Ch



IMA_READ_DATA_0 field descriptions

Field	Description
0–31 READ_0	Contains read data bits[31:0] to be written to RAM.

84.3.4 IMA module functional description

This section describes the operation of the IMA, beginning with the unlocking of reads and writes, the software initialization, then the software interface for generating single-bit ECC errors, generating multiple-bit ECC errors, and a software supported memory BIST.

84.3.4.1 Initialization sequence

This section describes which registers are reset due to hardware reset and what locations the user must initialize prior to doing an access by the IMA.

84.3.4.2 Hardware controlled initialization

In the IMA, registers and control logic are reset by hardware (system reset). A system reset deasserts output signals and resets general configuration bits.

84.3.4.3 User initialization (prior to asserting IMA_ENABLE[EN])

The user needs to initialize portions of the SoC prior to setting the IMA_ENABLE[EN] bit. The exact values depend on the particular application. Please consult the specific chapter's memory map to see how to turn off that peripheral so that accesses are not happening at the same time as the IMA. Without the peripheral shut down, there is a small chance that the IMA and the peripheral can interfere with each other in accessing the RAM.

84.3.4.3.1 Shut down the peripheral

The IMA can cause problems with the system or the peripheral if both the IMA and the other system components are accessing the RAM during normal operation.

To shut down the peripheral, please read the peripheral's documentation. The documentation will describe how to put the peripheral into a quiet state so that it will not access the RAM during the IMA access. If the other functional path is running at the same time, the data for the other peripheral will be corrupted and unknown.

84.3.4.3.2 Unlocking reads

The IMA will not allow a read to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the READ_KEY register to unlock writes. To unlock the read feature, the system will have to do the following:

1. Write 0xF06AB5BC to READ_KEY
2. Write 0x14081B56 to READ_KEY

The IMA has two features to make the READ_KEY lock in the following instance:

- Writing an incorrect key to READ_KEY

84.3.4.3.3 Unlocking writes

(The IMA will not allow a write to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the WRITE_KEY register to unlock writes. To unlock the write feature, the system will have to do the following:

1. Write 0x04A43F95 to WRITE_KEY
2. Write 0xE4A9EBF7 to WRITE_KEY

The IMA has a feature to make the WRITE_KEY lock in the following two instances:

1. Writing an incorrect key to WRITE_KEY
2. Not setting the IMA_ENABLE[bit] for 16,384 system clocks.

84.3.4.4 IMA lock

The IMA module also allows for read/write access locking via an input pin. If this input is asserted, both read/write accesses via IMA are locked. This locking scheme overrides the above KEY based locking scheme. Hence it provides additional safety over the KEY based locking method. However, usage of this feature is entirely SOC specific.

84.3.4.5 IMA access

There are two differences between a read access and a write access. The first difference is that the IMA_STATUS[WRITE_LOCK] bit has to be cleared for writes and the IMA_STATUS[READ_LOCK] has to be cleared for reads. The second difference is that the IMA_CTRL[READ] has to be set for reads and IMA_CTRL[READ] has to be cleared for writes.

The following sections shows the different accesses that are available depending on the options.

84.3.4.5.1 Read access

Read Accesses are to expected to follow the sequence below

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.

2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Set the READ bit in the IMA_CTRL register
4. Set the EN bit in the IMA_ENABLE register to initiate read access. The EN bit will self-clear after two clocks.
5. Software must wait an appropriate number of clocks depending on the speed of the SRAM for the access to complete.
6. Read data from READ_DATA registers. This causes data from RAM to be latched into the IMA and returned to the CPU on the next clock.

84.3.4.5.2 Write access

Write Accesses are to expected to follow the sequence below

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Write the Write Data in WRITE_DATA registers.
4. Clear the READ bit in the IMA_CTRL register
5. Set the EN bit in the IMA_ENABLE register to initiate write access. The EN bit will self-clear after two clocks.
6. Software must wait an appropriate number of clocks for the IMA write to occur, depending on the speed of the SRAM being written.

84.3.5 Application information

warning

The IMA will inhibit correct operation of the underlying peripherals. The application software is responsible for ensuring there is not a resource conflict.

The primary function of the IMA module is to provide a means for the system to program invalid ECC bits and see that the appropriate peripheral is able to detect the error. For a single-bit error, the peripheral should be able to correct the error. For a multiple-bit error, the peripheral should report an error.

Chapter 85
Fault Collection and Control Unit (FCCU)

85.1 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each RF:
    - No reset reaction
    - IRQ
    - Short functional reset
    - Long functional reset
    - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase
- The FCCU's configuration is lockable:

  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register.

- One of the error out pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.
- After power-on the error out pins have high impedance.[1] FCCU goes to operational state only on software request.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only). T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 16.67 ms.
- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 safe (IRCOSC) clock cycles**.**

Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when the system clock is faster than the IRC clock.

85.1.1 Acronyms and abbreviations

Table 85-1. Acronyms and abbreviations

Term	Description
UF	Unrecoverable fault
CPU	Central processing unit
EOUT	Error out
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRCOSC	Internal RC Oscillator
IRQ	Interrupt request
MC_ME	Mode entry module
RF	Recoverable fault

Table continues on the next page...

1. Actual value depends on device specific setting at pad level.

Table 85-1. Acronyms and abbreviations (continued)

Term	Description
NMI	Non-maskable interrupts
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

85.2 Main features

- Management of recoverable faults
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
  - long functional reset request pulse
  - short functional reset request pulse
  - NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock

85.3 Block diagram

The following figure is a top-level diagram of the FCCU module.

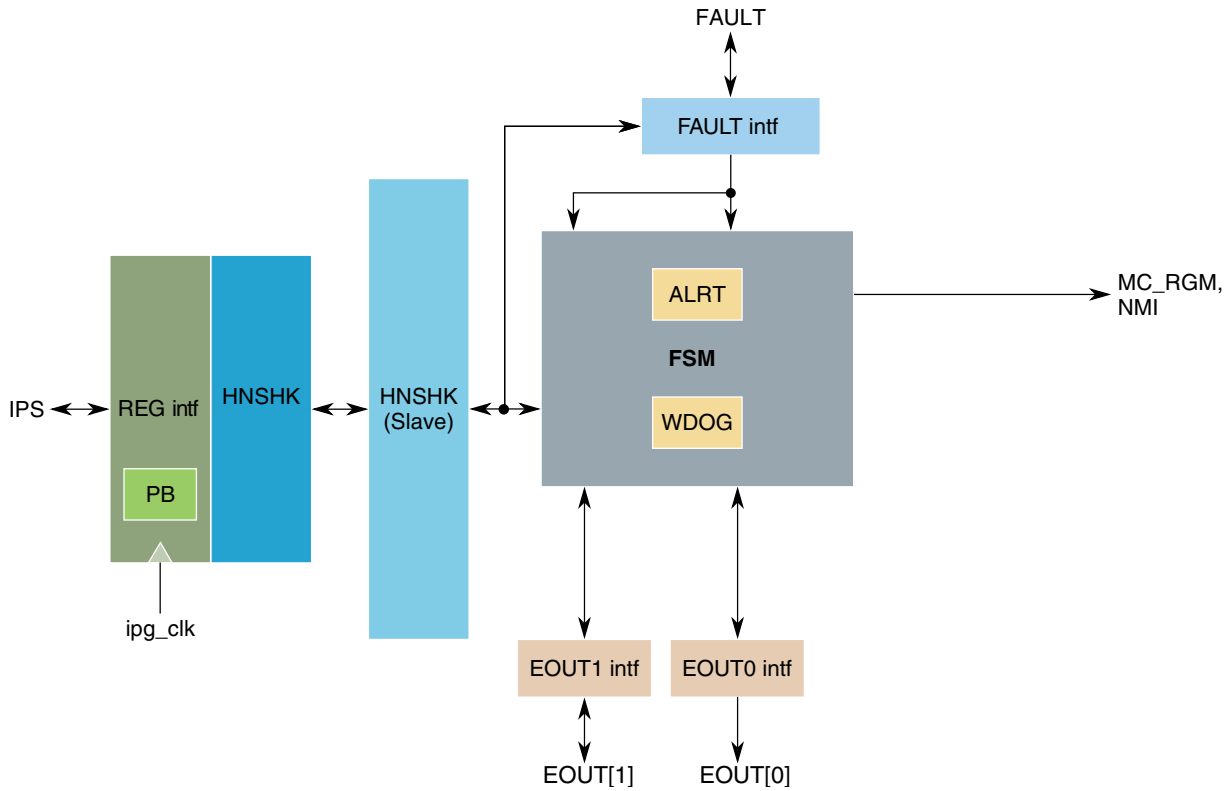


Figure 85-1. FCCU block diagram

The FCCU module includes the submodules listed in Table 85-2.

Table 85-2. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

85.4 Signal description

85.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in EOUT interface.

85.5 Functional description

85.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is asserted. That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is '0'. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched in an external module at the FCCU. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

The following types of reset are applicable:

- 'Destructive' reset: any type of reset related to a power failure condition that implies a complete system re-initialization
- Long 'functional' reset: implies the digital circuitry (most of it with some exceptions, such as the FCCU, STCU) initialization
- Short 'functional' reset: implies the digital circuitry (most of it with some exceptions, such as the FCCU, STCU) initialization.

For more information on each type of reset, see the MC_RGM and reset chapters.

85.5.2 FSM description

The FCCU module functionality is depicted by the FSM state diagram.

Basically four states are identified with the following meaning:

- CONFIG: The configuration state is used only to modify the default configuration of the FCCU. A sub-set of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, time-out, recoverable fault masking) can be accessed in write mode only in the CONFIG state.

  The CONFIG state is accessible only in NORMAL state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset of the FCCU, the transient lock register is unlocked by writing 0xBC into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register.

  After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

  The CONFIG to NORMAL state transition can be executed by SW or automatically following a time-out condition of the watchdog. In case the timeout information and the SW request to mode change to NORMAL appears at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in CONFIG mode) are reset to their default values. The movement to NORMAL is made.

  The incoming faults, occurring during the configuration phase (CONFIG state) are latched in order to process them when the FCCU is moved into the NORMAL state, according to the new configuration.

- NORMAL: This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:

  - unmasked recoverable faults with the time-out disabled → the FCCU moves to the FAULT state
  - unmasked recoverable faults with the time-out enabled → the FCCU moves to the ALARM state
  - masked recoverable faults → the FCCU stays in NORMAL state

- ALARM: the FCCU moves into the ALARM state when an unmasked fault occurs and the time-out is enabled. The transition to the ALARM state goes along with an interrupt ALARM, if enabled. By definition, this fault may be recovered within a programmable time-out period, before it generates a transition to the FAULT state. The time-out is re-initialized if the FCCU state moves to the NORMAL state. The time-out restarts following the recovery from the FAULT state.
- FAULT: the FCCU moves into the FAULT state when one of the following condition occurs:
  - time-out related to a recoverable fault when the FCCU is in the ALARM state
  - unmasked recoverable faults with the time-out disabled

The transition from NORMAL/ALARM state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)
- Non Maskable Interrupt (NMI) is routed only to the Safety Core

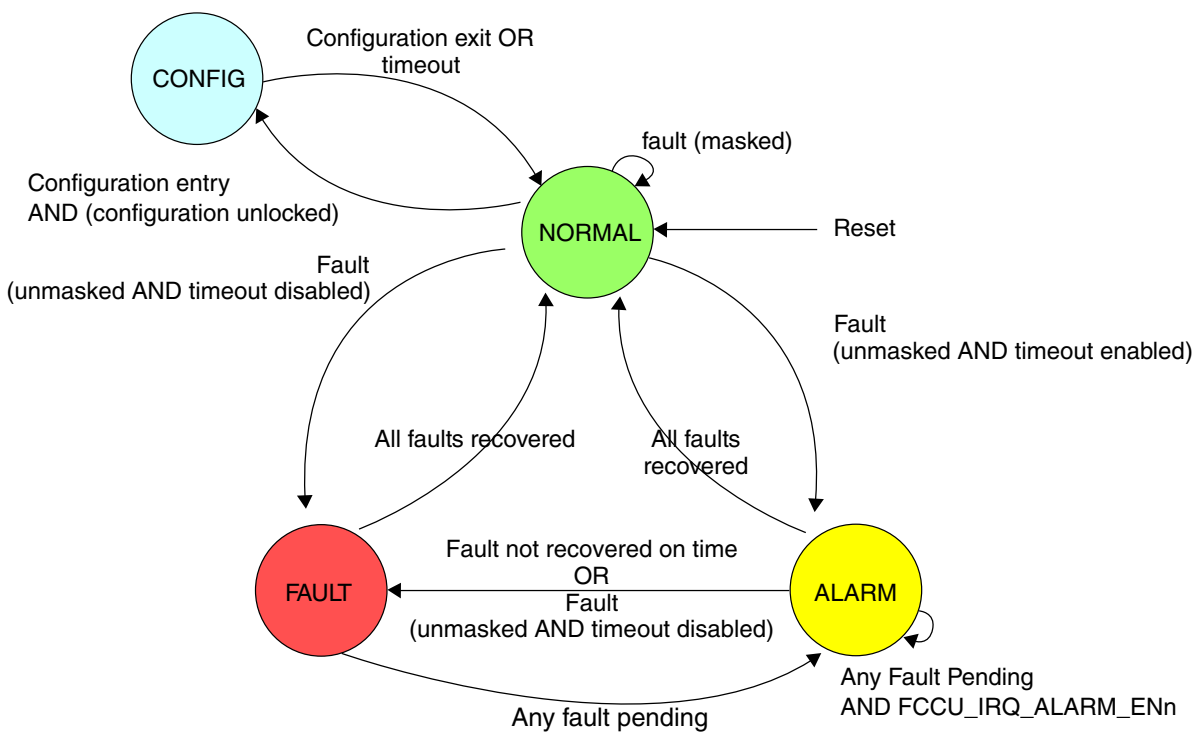


Figure 85-2. FCCU state diagram

85.5.3 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (recoverable) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a recoverable fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the recoverable faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the recoverable faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any recoverable fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the recoverable faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

85.5.4 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a recoverable FAULT management (ALARM state), see Figure 85-3 and Figure 85-4, is as follows:

- recoverable fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)

- FCCU state transition ALARM → NORMAL

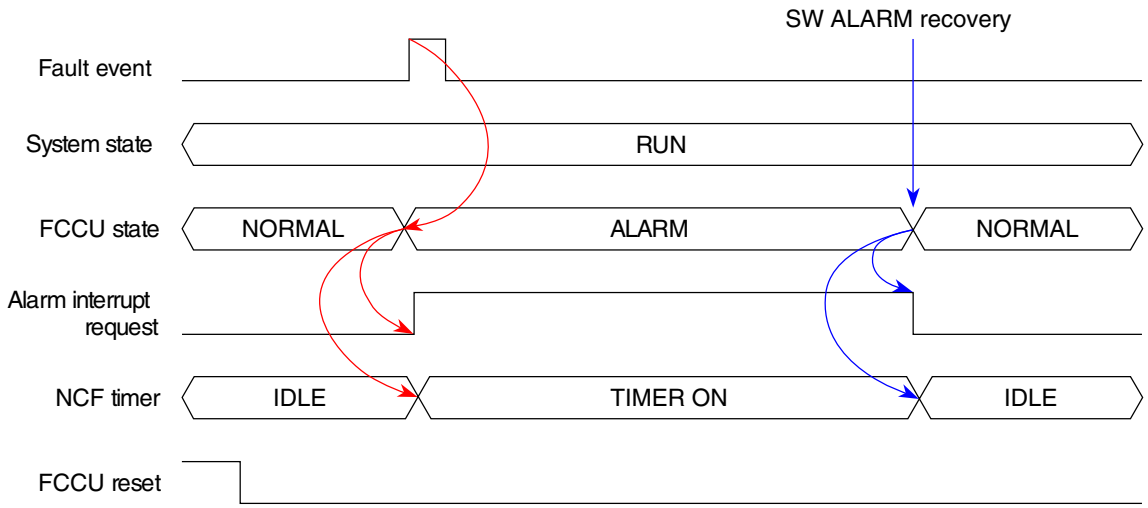


Figure 85-3. recoverable FAULT (ALARM state) recovery (a)

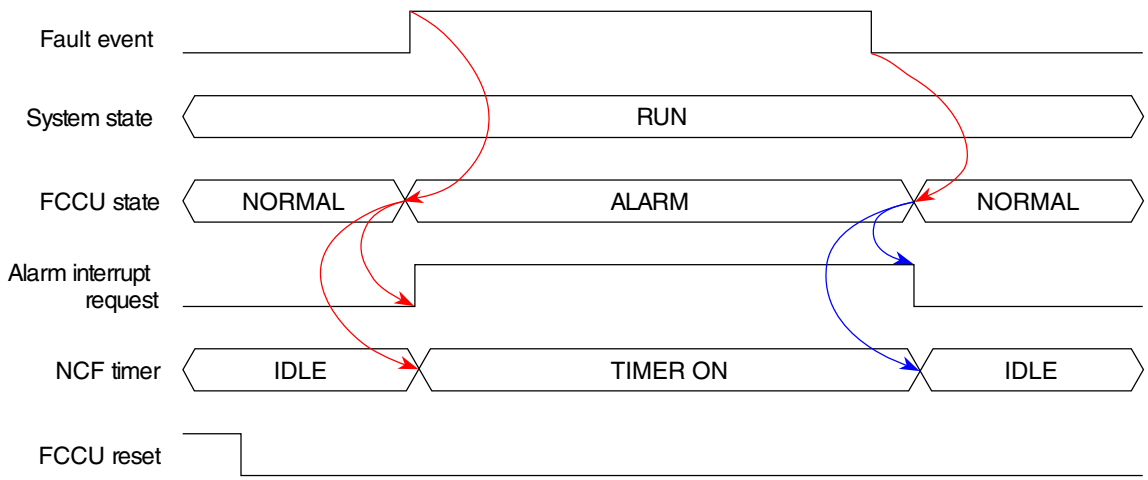


Figure 85-4. recoverable FAULT (ALARM state) recovery (b)

A typical sequence related to a recoverable FAULT management (ALARM → FAULT state), see Figure 85-5, is as follows:

- recoverable fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT
  - NMI assertion (if enabled)

- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)
  - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
  - system state transition: SAFE → RUN

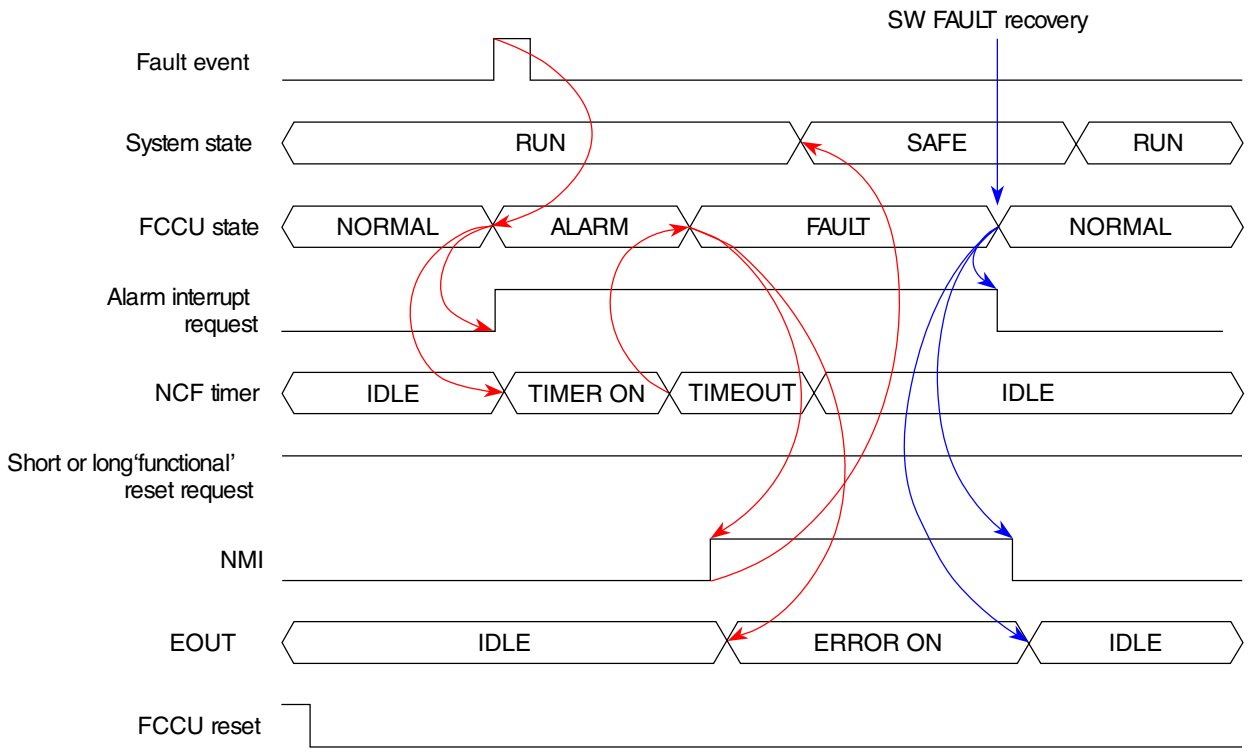


Figure 85-5. FAULT (ALARM → FAULT state) recovery

85.5.5 NMI/WKPU interface

The NMI signal internally generated by FCCU is masked when:

- FCCU asynchronous reset is in progress, or
- Chip mode = RESET

and unmasked when:

- a SW change request of the chip mode is triggered

NOTE

The unmasked, internally generated NMI signal is applied to all CPU cores.

The following figure shows the logical scheme.

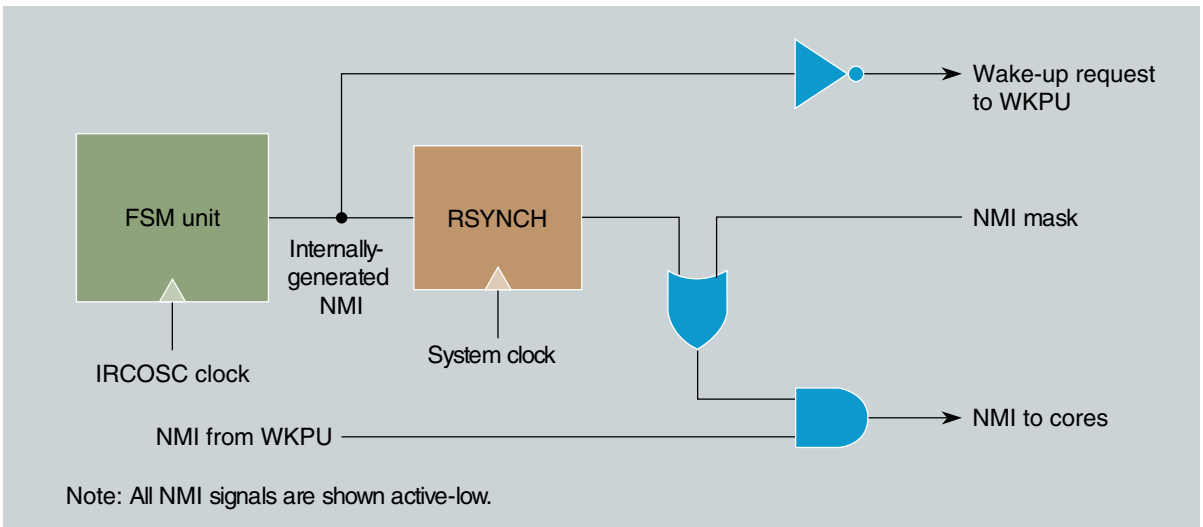


Figure 85-6. NMI/WKPU scheme

85.5.6 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

  The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).

- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.

3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR. This feature is optionally programmable inside the STCU.

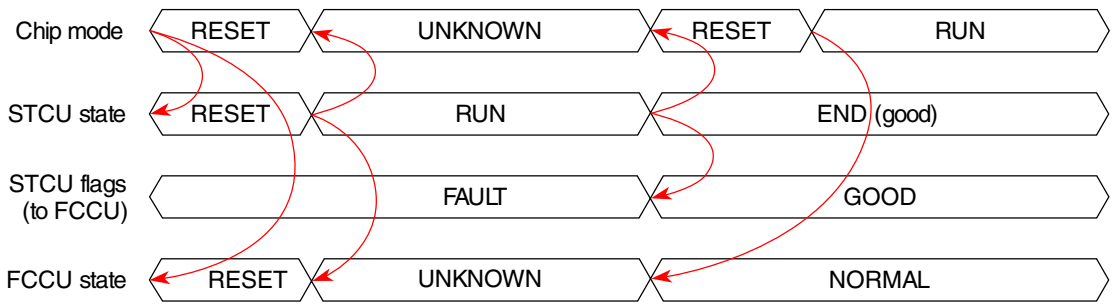


Figure 85-7. STCU-FCCU (case 1)

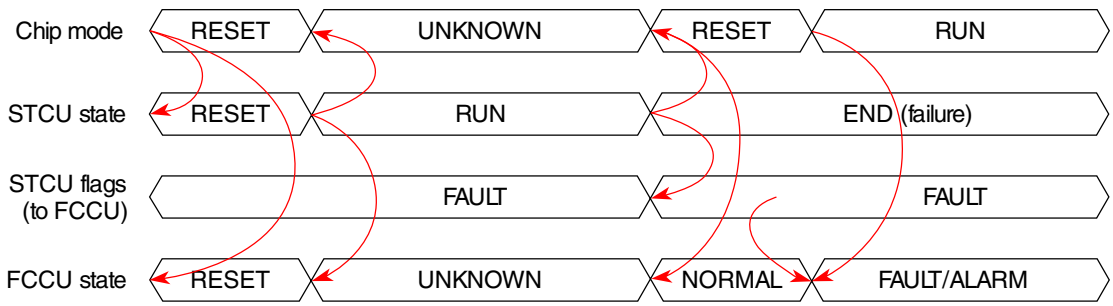


Figure 85-8. STCU-FCCU (case 2)

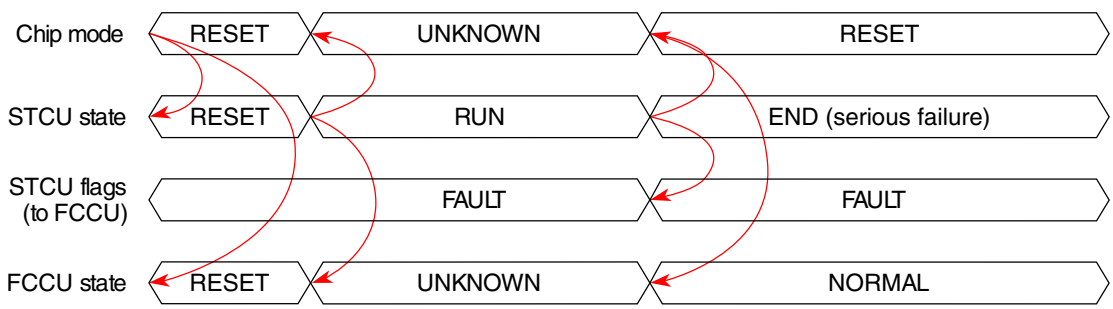


Figure 85-9. STCU-FCCU (case 3)

85.5.7 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT interface) as a failure indication to the external world. These signals need to be mapped to appropriate pins for external visibility. Since SIUL2 does not control the pad buffer enable during Alternate function, FCCU provides a mechanism to drive the pad in OD (Open Drain) mode where the pad is driven Hi-Z only when FCCU drives logic '1' else it drives logic '0' (when

FCCU drives logic '0'). The selection to OD is done by OD bit in FCCU_CFG register.Different protocols for the EOUT interface are supported, selecting the FCCU_CFG.FOM register field:

- Dual-rail protocol
- Time switching protocol
- Bi-stable protocol
- Test mode

The signal polarity can be programmed by setting the PS field in the FCCU_CFG register. For example, for PS = 0, EOUT[0] = 0 and EOUT[1] = 1 in FAULT state for bi-stable and for PS = 1, it is inverted. All the diagrams and tables are related to the default configuration selection: switching mode (FCCU_CFG.SM = 0). In case of inverted polarity (FCCU_CFG.PS = 1) all the values on the EOUT output pins are inverted.

Two modes can be programmed to define the EOUT protocol transitions in dual-rail or time-switching mode:

- slow switching mode: no EOUT frequency violation during the FCCU state transition (NORMAL to FAULT or viceversa and CONFIG to NORMAL). The EOUT protocol transition occurs after a max delay equal to the duration of the semi-period of the EOUT frequency.
- fast switching mode: The EOUT protocol transition (NORMAL to FAULT or viceversa and CONFIG to NORMAL) occurs immediately. A pulse with the minimum duration corresponding to 16 MHz/1024 (IRCOSC clock) period can occur in fast switching mode. It implies a frequency violation of the EOUT protocol.

With an IRCOSC clock of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and the IRCOSC clock detecting the edge transition of the EOUT protocol in dual-rail or time-switching mode.

In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before. If SW configures the error pins to OK(1) and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and error pins indicate NOK, such as an incoming fault is not masked when SW has set the error pin to high. During the T_min by a non-SW fault, the FCCU FSM moves independently of this pin state (low) and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself

in and it is not possible to set the pins to OK by SW moving FCCU to CONFIG state, as long as this timer is running. No SW intervention is needed to bring the pin from the low state.

The SW can bring the pin back to OK state by clearing the faults and waiting for the T_min to expire, after which the FCCU automatically enters NORMAL state and the error pin indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

These resets influence the state of the failure indication pins:

- Power on reset
- Destructive reset
- FOSU time-out reset
- Pin reset (RESET_b)

Other resets should not influence the state of the failure indication pins.

NOTE

The transition from fault to config state is not possible but is shown to display the behavior in all four states — reset, normal, error, and config.

NOTE

FCCU remains in NORMAL state when fault is disabled; hence, "NA" (Not Applicable) appears in the following table.

NOTE

The following table is valid only after SW has written fccu_set_after_reset bit to 1.

Table 85-3. FCCU error pin behavior on STATE transitions

Fault signaling state	Transition from NORMAL to FAULT state			NORMAL+CONFIG states			Transition from NORMAL to ALARM state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

NOTE

During first configuration phase, EOUT pads will be hi-z state and thereafter, it will follow the configured EOUT protocol encoding.

85.5.7.1 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 85-4 and the related timing diagram is given in Figure 85-10 and Figure 85-11.

Table 85-4. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	
reset	Hi impedence[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on the SoC setting at pad level.

As long as FCCU is in NORMAL or ALARM state, output will show "non-faulty" signal. Output pins EOUT[0:1] will toggle between 01 and 10 with a given frequency of 61 Hz.

In the RESET phase the output pins are set as "high impedance".

Note

Figure 85-10 and Figure 85-11 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

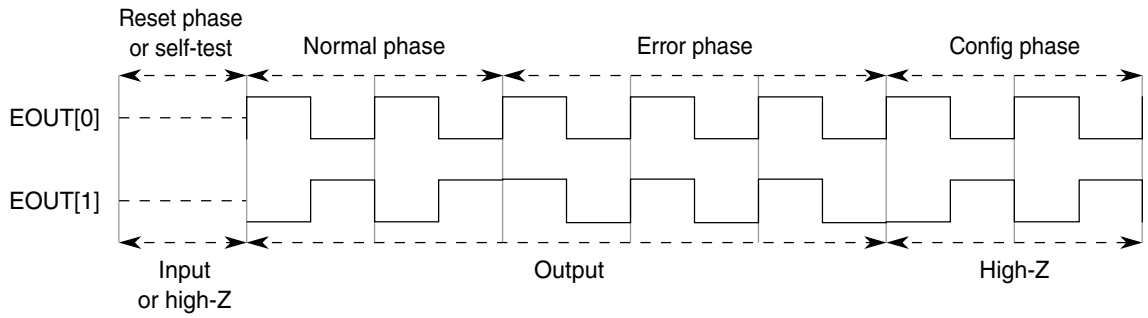


Figure 85-10. Dual-rail protocol (slow switching mode)

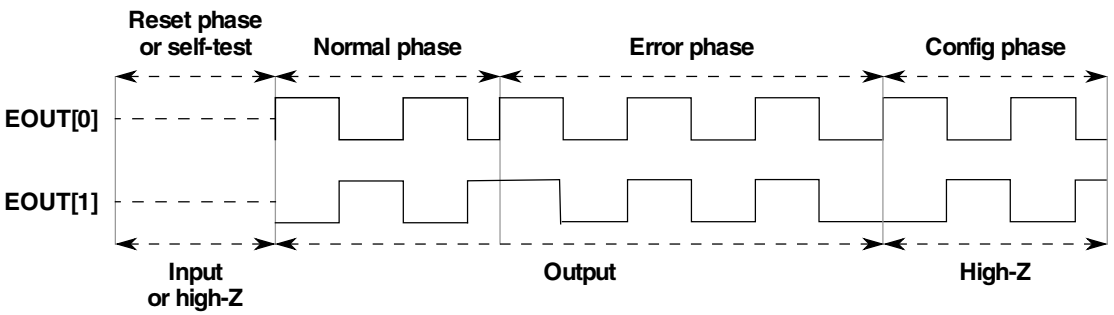


Figure 85-11. Dual-rail protocol (fast switching mode)

85.5.7.2 Time-switching protocol

The encoding scheme is given in Table 85-5 and the related timing diagram is given in Figure 85-12.

Table 85-5. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency of 61 Hz.

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[2]

Note

Figure 85-12 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

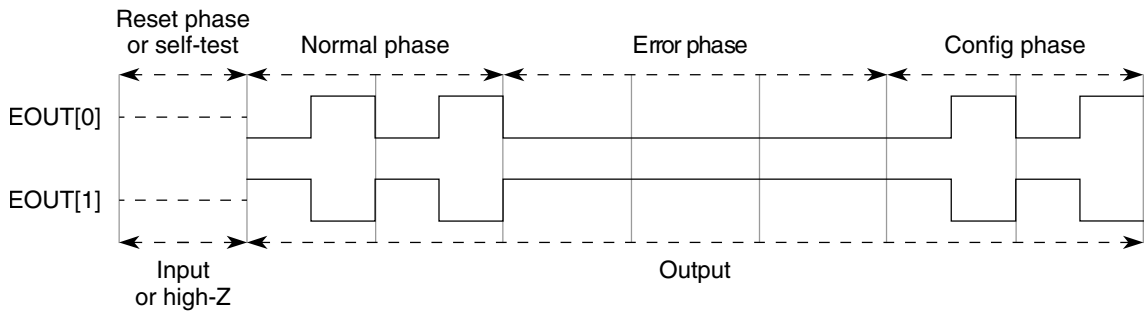


Figure 85-12. Time-switching protocol

85.5.7.3 Bi-stable protocol

The encoding scheme is given in Table 85-6 and the related timing diagram is given in Figure 85-13.

Table 85-6. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	no toggling

1. Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[3]

2. Actual value depends on device specific settings at pad level.
3. Actual value depends on device specific settings at pad level.

Note

Figure 85-13 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

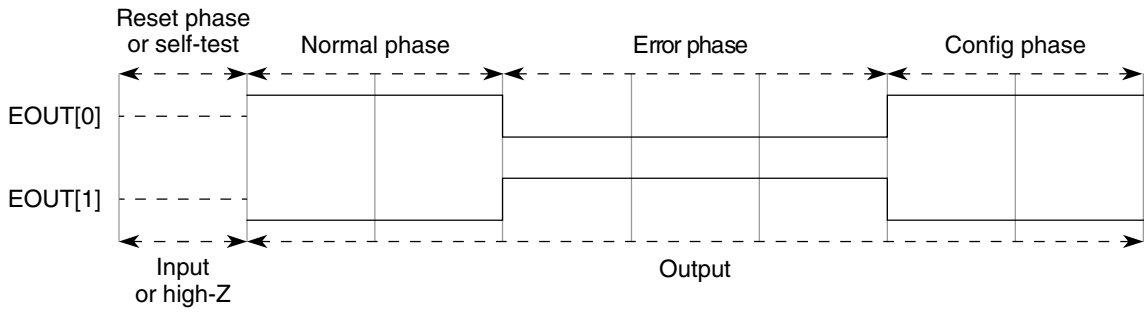


Figure 85-13. Bi-stable protocol

85.5.8 Error signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

85.6 Register description

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

The FCCU generates an access error in the following cases:

- any write/read access to a reserved address
- any write access to the configuration registers not executed in the CONFIG state
- any write access to the Transient and Permanent Lock Register in any mode other than supervisor access mode

With the exception noted above, the FCCU registers are accessible (read/write) in both user and supervisor mode.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible RF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

FCCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (FCCU_CTRL)	32	R/W	0000_00C0h	85.6.1/4415
4	CTRL Key Register (FCCU_CTRLK)	32	W	0000_0000h	85.6.2/4418
8	Configuration Register (FCCU_CFG)	32	R/W	See section	85.6.3/4419
1C	RF Configuration Register (FCCU_RF_CFG0)	32	R/W	Undefined	85.6.4/4421
20	RF Configuration Register (FCCU_RF_CFG1)	32	R/W	Undefined	85.6.4/4421
24	RF Configuration Register (FCCU_RF_CFG2)	32	R/W	Undefined	85.6.4/4421
28	RF Configuration Register (FCCU_RF_CFG3)	32	R/W	Undefined	85.6.4/4421
4C	RFS Configuration Register (FCCU_RFS_CFG0)	32	R/W	See section	85.6.5/4422
50	RFS Configuration Register (FCCU_RFS_CFG1)	32	R/W	See section	85.6.5/4422
54	RFS Configuration Register (FCCU_RFS_CFG2)	32	R/W	See section	85.6.5/4422
58	RFS Configuration Register (FCCU_RFS_CFG3)	32	R/W	See section	85.6.5/4422
5C	RFS Configuration Register (FCCU_RFS_CFG4)	32	R/W	See section	85.6.5/4422
60	RFS Configuration Register (FCCU_RFS_CFG5)	32	R/W	See section	85.6.5/4422
64	RFS Configuration Register (FCCU_RFS_CFG6)	32	R/W	See section	85.6.5/4422
68	RFS Configuration Register (FCCU_RFS_CFG7)	32	R/W	See section	85.6.5/4422
80	UF Status Register (FCCU_RF_S0)	32	R/W	0000_0000h	85.6.6/4423
84	UF Status Register (FCCU_RF_S1)	32	R/W	0000_0000h	85.6.6/4423
88	UF Status Register (FCCU_RF_S2)	32	R/W	0000_0000h	85.6.6/4423
8C	UF Status Register (FCCU_RF_S3)	32	R/W	0000_0000h	85.6.6/4423
90	RF Key Register (FCCU_RFK)	32	W	0000_0000h	85.6.7/4425
94	RF Enable Register (FCCU_RF_E0)	32	R/W	See section	85.6.8/4426
98	RF Enable Register (FCCU_RF_E1)	32	R/W	See section	85.6.8/4426
9C	RF Enable Register (FCCU_RF_E2)	32	R/W	See section	85.6.8/4426
A0	RF Enable Register (FCCU_RF_E3)	32	R/W	See section	85.6.8/4426

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	RF Time-out Enable Register (FCCU_RF_TOE0)	32	R	See section	85.6.9/4427
A8	RF Time-out Enable Register (FCCU_RF_TOE1)	32	R	See section	85.6.9/4427
AC	RF Time-out Enable Register (FCCU_RF_TOE2)	32	R	See section	85.6.9/4427
B0	RF Time-out Enable Register (FCCU_RF_TOE3)	32	R	See section	85.6.9/4427
B4	RF Time-out Register (FCCU_RF_TO)	32	R	0000_FFFFh	85.6.10/ 4428
B8	CFG Timeout Register (FCCU_CFG_TO)	32	R/W	0000_0006h	85.6.11/ 4429
BC	IO Control Register (FCCU_EINOUT)	32	R/W	0000_0000h	85.6.12/ 4430
C0	Status Register (FCCU_STAT)	32	R	0000_0010h	85.6.13/ 4432
C4	NA Freeze Status Register (FCCU_N2AF_STATUS)	32	R	0000_0000h	85.6.14/ 4434
C8	AF Freeze Status Register (FCCU_A2FF_STATUS)	32	R/W	0000_0000h	85.6.15/ 4435
CC	NF Freeze Status Register (FCCU_N2FF_STATUS)	32	R/W	0000_0000h	85.6.16/ 4436
D0	FA Freeze Status Register (FCCU_F2A_STATUS)	32	R	0000_0000h	85.6.17/ 4437
DC	RF Fake Register (FCCU_RFF)	32	R/W	0000_0000h	85.6.18/ 4438
E0	IRQ Status Register (FCCU_IRQ_STAT)	32	R/W	0000_0000h	85.6.19/ 4439
E4	IRQ Enable Register (FCCU_IRQ_EN)	32	R/W	0000_0000h	85.6.20/ 4440
E8	XTMR Register (FCCU_XTMR)	32	R	0000_0000h	85.6.21/ 4441
EC	MCS Register (FCCU_MCS)	32	R	See section	85.6.22/ 4442
F0	Transient Lock Register (FCCU_TRANS_LOCK)	32	W	0000_0000h	85.6.23/ 4445
F4	Permanent Lock Register (FCCU_PERMNT_LOCK)	32	W	0000_0000h	85.6.24/ 4445
F8	Delta T Register (FCCU_DELTA_T)	32	R/W	0000_0000h	85.6.25/ 4446
FC	IRQ Alarm Enable Register (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	85.6.26/ 4447
100	IRQ Alarm Enable Register (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	85.6.26/ 4447
104	IRQ Alarm Enable Register (FCCU_IRQ_ALARM_EN2)	32	R/W	0000_0000h	85.6.26/ 4447
108	IRQ Alarm Enable Register (FCCU_IRQ_ALARM_EN3)	32	R/W	0000_0000h	85.6.26/ 4447

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C	NMI Enable Register (FCCU_NMI_EN0)	32	R/W	0000_0000h	85.6.27/ 4447
110	NMI Enable Register (FCCU_NMI_EN1)	32	R/W	0000_0000h	85.6.27/ 4447
114	NMI Enable Register (FCCU_NMI_EN2)	32	R/W	0000_0000h	85.6.27/ 4447
118	NMI Enable Register (FCCU_NMI_EN3)	32	R/W	0000_0000h	85.6.27/ 4447
11C	EOUT Signaling Enable Register (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	85.6.28/ 4448
120	EOUT Signaling Enable Register (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	85.6.28/ 4448
124	EOUT Signaling Enable Register (FCCU_EOUT_SIG_EN2)	32	R/W	0000_0000h	85.6.28/ 4448
128	EOUT Signaling Enable Register (FCCU_EOUT_SIG_EN3)	32	R/W	0000_0000h	85.6.28/ 4448

85.6.1 Control Register (FCCU_CTRL)

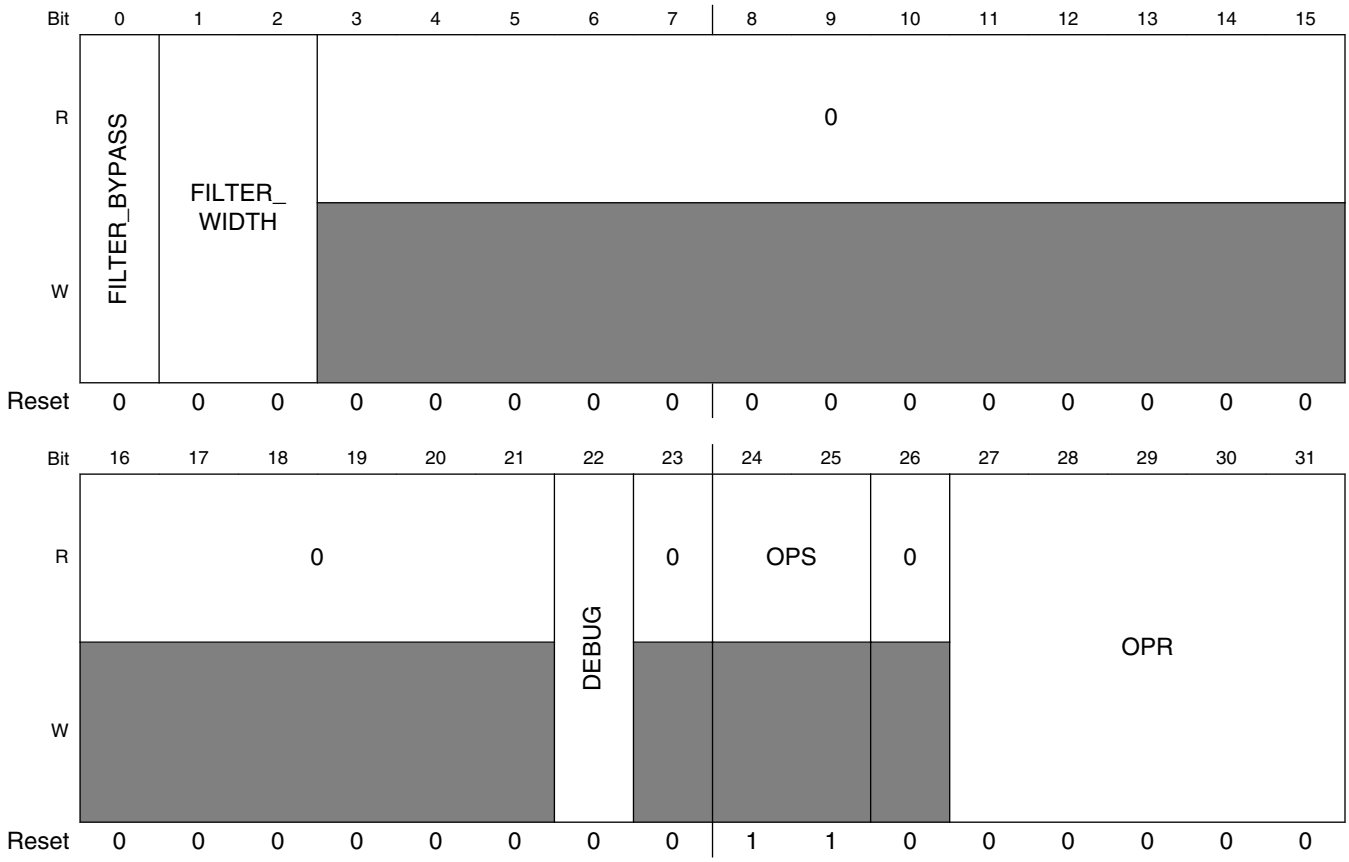
The FCCU_CTRL register allows the following operations to be executed:

- moving the FCCU state from the NORMAL state into the CONFIG state
- moving the FCCU state from the CONFIG state into the NORMAL state
- reading or to clear the RF status register
- reading the FCCU FSM status register
- reading or to clear the FCCU freeze registers
- reading the ALARM timer
- reading the Watchdog timer

Some critical operations require a key as defined in the FCCU_CTRLK register.

Register description

Address: 0h base + 0h offset = 0h



FCCU_CTRL field descriptions

Field	Description
0 FILTER_ BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1–2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 µs 01 filters glitches up to 75 µs 10 filters glitches up to 100 µs 11 filters glitches up to 100 µs
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug mode entry When debug signal is asserted and this bit is set, FCCU moves into debug state and FCCU's FSM remains in the same state. 0 Normal operation 1 Put FCCU into debug mode
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation run. The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP12 must not be programmed. The OPR field is automatically set to OP12 when the FCCU_RF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. The operations OP21-OP30 are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: • wrong access (missing or wrong key) to the FCCU_RF_S register (clear operation OP12) • wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) • OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (see the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (see the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (see the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (see the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (see the F2AF_STATUS register) [OP7] 01000 Reserved 01001 Reserved 01010 Read the RF status register (see the FCCU_RF_S register) [OP10] 01011 Reserved 01100 RF status clear operation in progress (see the FCCU_RF_S register) [OP12] 01101 Clear the freeze status registers (see the freeze registers) [OP13] 01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111 Clear the operation status (OPS=Idle) [OP15] 10000 Reserved 10001 Read the ALARM timer (see the FCCU_XTMR register) [OP17] 10010 Reserved

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	10011 Read the CFG timer (see the FCCU_XTMR register) [OP19] 10100 Read the Error Pin low counter value (see XTMR Register (FCCU_XTMR)) [OP20] 10101-11110 Forbidden 11111 Reserved

85.6.2 CTRL Key Register (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

NOTE

The FCCU_CTRLK and FCCU_CTRL registers must be written with consecutive instructions. Both registers must be written as 32-bit values. Do not use read-modify-write instructions, such as bit field instructions, to modify these registers.

The FCCU_CTRLK register is not readable, a 0000_0000h value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	CTRLK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Control register key. = 913756AFh: Key for the operation OP1 = 825A132Bh: Key for the operation OP2

85.6.3 Configuration Register (FCCU_CFG)

The FCCU_CFG register defines the global configuration for the FCCU module.

NOTE

This register is writable only when the FCCU is in the CONFIG state.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R								FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		0		0			
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	Reserved		OD	0	SM	PS	FOM			Reserved					
W																
Reset	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*

* Notes:

- Reserved field: ("Reserved") Read-Only (RO)

FCCU_CFG field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 FCCU_SET_ AFTER_RESET	This bit controls the enable of the o/p error pin after reset lifts. After power-on the error out pins shall be in high impedance. **NOTE:** Actual value depends on the SoC settings at pad level. They will go to normal state only on software request, that is, this bit is set to '1'. It is SW's responsibility to write this bit to 1 else the error pin stays in High-Z state after reset lifts and FCCU is not functioning. 0 FCCU's Error indication stops functioning and error pins are in HI-Z state. (note: actual value depends on the SoC settings at pad level) 1 Write to start FCCU functioning.
8–9 FCCU_SET_ CLEAR	Error pin state can be controlled by these bits These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z. **NOTE:** Actual value depends on the SoC settings at pad level. Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves CONFIG state. After this is in effect, software maintains priority over FCCU FSM except for the case of '11' mentioned above. The switch back to FSM control is again done by writing into these fields with 00 or 10 or by going to CONFIG state. 00 FCCU acts independent of above SW control 01 Error Pin goes in Faulty state. 10 FCCU acts independent of above SW control. 11 write: Error Pin goes OK state (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault).
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved.
17–18 Reserved	This field is reserved.
19 OD	Open Drain Mechanism to select between Push-pull and Open drain(OD) mode for the error indicating pin(s) 0 push-pull 1 OD
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SM	Switching mode. SM has no effect on the bi-stable protocol. 0 EOUT protocol (dual-rail, time-switching) slow switching mode. 1 EOUT protocol (dual-rail, time-switching) fast switching mode.
22 PS	Polarity selection

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	This bit can be read and written by the software. Active state here means a state indicating FAULT and applicable when pins are non-toggling in Error phase. 0 EOUT[1] active high, EOUT[0] active low 1 EOUT[1] active low, EOUT[0] active high
23–25 FOM	Fault Output Mode selection. **NOTE:** In Test *x* mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/IRCOSC clock. 000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time Switching [EOUT[1:0]= output to be used] 010 Bi-Stable 011 Reserved 100 Reserved 101 Test0 (controlled by FCCU_EOUT register) [EOUT[0]= input, EOUT[1]= output] 110 Test1 (controlled by FCCU_EOUT register) [EOUT[0]= output, EOUT[1]= output] 111 Test2 (controlled by FCCU_EOUT register) [EOUT[0]= output, EOUT[1]= input]
26–31 Reserved	This field is reserved.

85.6.4 RF Configuration Register (FCCU_RF_CFG*n*)

NOTE

The reset value is chip-specific. See the chip-specific FCCU information.

FCCU_RF_CFGx registers contain the configuration of each recoverable fault in terms of fault recovery management. The configuration depends on the type of signaling of a fault event. HW recoverable faults should be configured only if a previous latching stage captures and hold the physical fault otherwise the fault can be lost. All the other faults should be configured as SW fault.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 85-7 shows FCCU RF configuration register channels.

Table 85-7. FCCU RF configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
1Ch	FCCU_RF_CFG0	RFC[31:0]
20h	FCCU_RF_CFG1	RFC[63:32]

Table continues on the next page...

Table 85-7. FCCU RF configuration register channels (continued)

Address offset	Register name	Channel range (x) (bit location [0:31])
24h	FCCU_RF_CFG2	RFC[95:64]
28h	FCCU_RF_CFG3	RFC[127:96]

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FCCU_RF_CFG*n* field descriptions

Field	Description
0–31 RFCx	Recovereable fault configuration. The recoverable fault configuration defines the fault recovery mode. HW recoverable faults are self recovered (status flag clearing and related) if the root cause (input fault) has been removed. SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag. See Table 85-7 for register offset to channel number relationship. 0 HW recoverable fault 1 SW recoverable fault

85.6.5 RFS Configuration Register (FCCU_RFS_CFG*n*)

NOTE

The reset value is chip-specific. See the chip-specific FCCU information.

The FCCU_RFS_CFGx registers contain the configuration of each recoverable fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Table 85-8 shows FCCU RFSNCFS configuration register channels.

Table 85-8. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_RFS_CFG0	RFSCx[15:0]
50h	FCCU_RFS_CFG1	RFSCx[31:16]
54h	FCCU_RFS_CFG2	RFSCx[47:32]
58h	FCCU_RFS_CFG3	RFSCx[63:48]
5Ch	FCCU_RFS_CFG4	RFSCx[79:64]
60h	FCCU_RFS_CFG5	RFSCx[95:80]
64h	FCCU_RFS_CFG6	RFSCx[111:96]
68h	FCCU_RFS_CFG7	RFSCx[127:112]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- See the chip-specific FCCU information.x = Undefined at reset.

FCCU_RFS_CFG*n* field descriptions

Field	Description
0–31 RFSCx	Recovereable fault state configuration See Table 85-8 for register offset to channel number relationship. **NOTE:** The reset value is chip-specific. See the chip-specific FCCU information for details. 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

85.6.6 UF Status Register (FCCU_RF_S*n*)

The FCCU_RF_Sx register contains the latched fault indication collected from the recovereable fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the RF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_RF_Ex register and the respective enable bit for the time-out is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_RF_Ex register and the respective enable bit for the time-out is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the time-out (FCCU_TO register) is elapsed before recovering from the fault.

The time-out is stopped only when the FCCU returns in the NORMAL state.

The FCCU_RF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_RF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_RFK register.
2. Clear the status (flag) bit RFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_RF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_RF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_RF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_RF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_RFK register
- to attempt to clear a HW recoverable fault

Table 85-9 shows FCCU RF status register channels.

Table 85-9. FCCU RF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
80h	FCCU_RF_S0	RFS[31:0]
84h	FCCU_RF_S1	RFS[63:32]
88h	FCCU_RF_S2	RFS[95:64]
8Ch	FCCU_RF_S3	RFS[127:96]

Address: 0h base + 80h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_RF_S*n* field descriptions

Field	Description
0–31 RFSx	Recoverable fault status. The status bits related to the recoverable fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. See Table 85-9 for register offset to channel number relationship. 0 No "recoverable" fault latched 1 "Recoverable" fault latched

85.6.7 RF Key Register (FCCU_RFK)

The FCCU_RFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_RF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

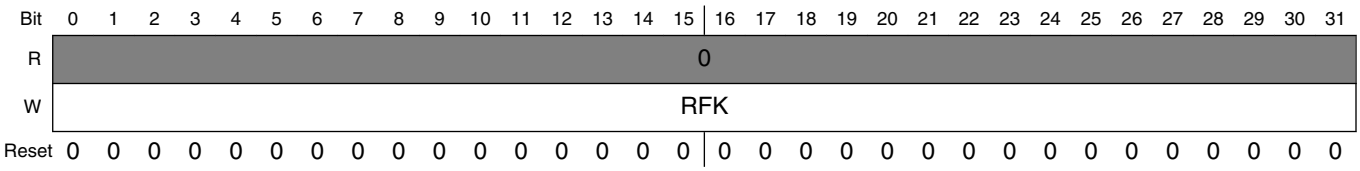
1. Write the key into the FCCU_RFK register.

2. Clear the status (flag) bit RFSx .

The key must be written for each FCCU_RF_Sx clear operation.

The FCCU_RFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h



FCCU_RFK field descriptions

Field	Description
0–31 RFK	recoverable fault key = AB34_98FEh

85.6.8 RF Enable Register (FCCU_RF_E*n*)

NOTE

The reset value is chip-specific. See the chip-specific FCCU information.

The FCCU_RF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_RF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Table 85-10 shows FCCU RFE enable register channels.

Table 85-10. FCCU RF enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
94h	FCCU_RF_E0	RFE[31:0]
98h	FCCU_RF_E1	RFE[63:32]
9Ch	FCCU_RF_E2	RFE[95:64]
A0h	FCCU_RF_E3	RFE[127:96]

Address: 0h base + 94h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- See the chip-specific FCCU information.x = Undefined at reset.

FCCU_RF_E*n* field descriptions

Field	Description
0–31 RFEx	Recoverable fault enable See Table 85-10 for register offset to channel number relationship. **NOTE:** The reset value is chip specific. See the chip-specific FCCU information for details. 0 No actions following the respective recoverable fault assertion 1 FCCU moves to ALARM or FAULT state

85.6.9 RF Time-out Enable Register (FCCU_RF_TOE*n*)

NOTE

The reset value is chip-specific. See the chip-specific FCCU information.

The FCCU_RF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective recoverable fault is enabled (RFEx and RFTOEx are set). In case the respective time-out is disabled (RFTOEx is cleared) and the recoverable fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related recovereable fault is asserted. The timer (preset with the time-out value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the time-out the FCCU moves from the ALARM state to the FAULT state.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 85-11 shows FCCU RF timeout enable register channels.

Table 85-11. FCCU RF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
A4h	FCCU_RF_TOE0	RFTOE[31:0]
A8h	FCCU_RF_TOE1	RFTOE[63:32]
ACh	FCCU_RF_TOE2	RFTOE[95:64]
B0h	FCCU_RF_TOE3	RFTOE[127:96]

Address: 0h base + A4h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFTOEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- See the chip-specific FCCU information.x = Undefined at reset.

FCCU_RF_TOE*n* field descriptions

Field	Description
0–31 RFTOEx	Fault time-out enable See Table 85-11 for register offset to channel number relationship. **NOTE:** The reset value is chip specific. See the chip-specific FCCU information for details. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

85.6.10 RF Time-out Register (FCCU_RF_TO)

The RF_TO register defines the preset value of the timer for the recovery of the recoverable faults (enabled). Once FCCU enters in ALARM state, following the assertion of a recoverable fault enabled (RFEx and RFTOEx are set), the timer starts the count down.

If the fault is not recovered within the time-out the FCCU moves from the ALARM state to the FAULT state. The alarm time-outs value should be programmed less than the FOSU_COUNT, elsedestructive resets may be generated by FOSU time-out.

NOTE

This register is writable only when the FCCU is in the CONFIG state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FCCU_RF_TO field descriptions

Field	Description
0–31 TO	Recoverable fault timeout Timeout = (TO) × $T_{RC16MHz}$

85.6.11 CFG Timeout Register (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the time-out, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. The description for a configuration register will contain a NOTE that it is writable only in the CONFIG state. The watchdog time-out is clocked with the IRCOSC clock (16 MHz). The default time-out value is 4.096 ms.Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

NOTE

The FCCU_CFG_TO register is writable in any state excluding the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog time-out the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													TO		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration time-out Timeout = $T_{RC16MHz} \times 2^{(TO + 10)}$ 000 Time-out = 64 µs ... 111 Time-out = 8.192 ms

85.6.12 IO Control Register (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)
- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

Table 85-12 shows Bi-stable encoding.

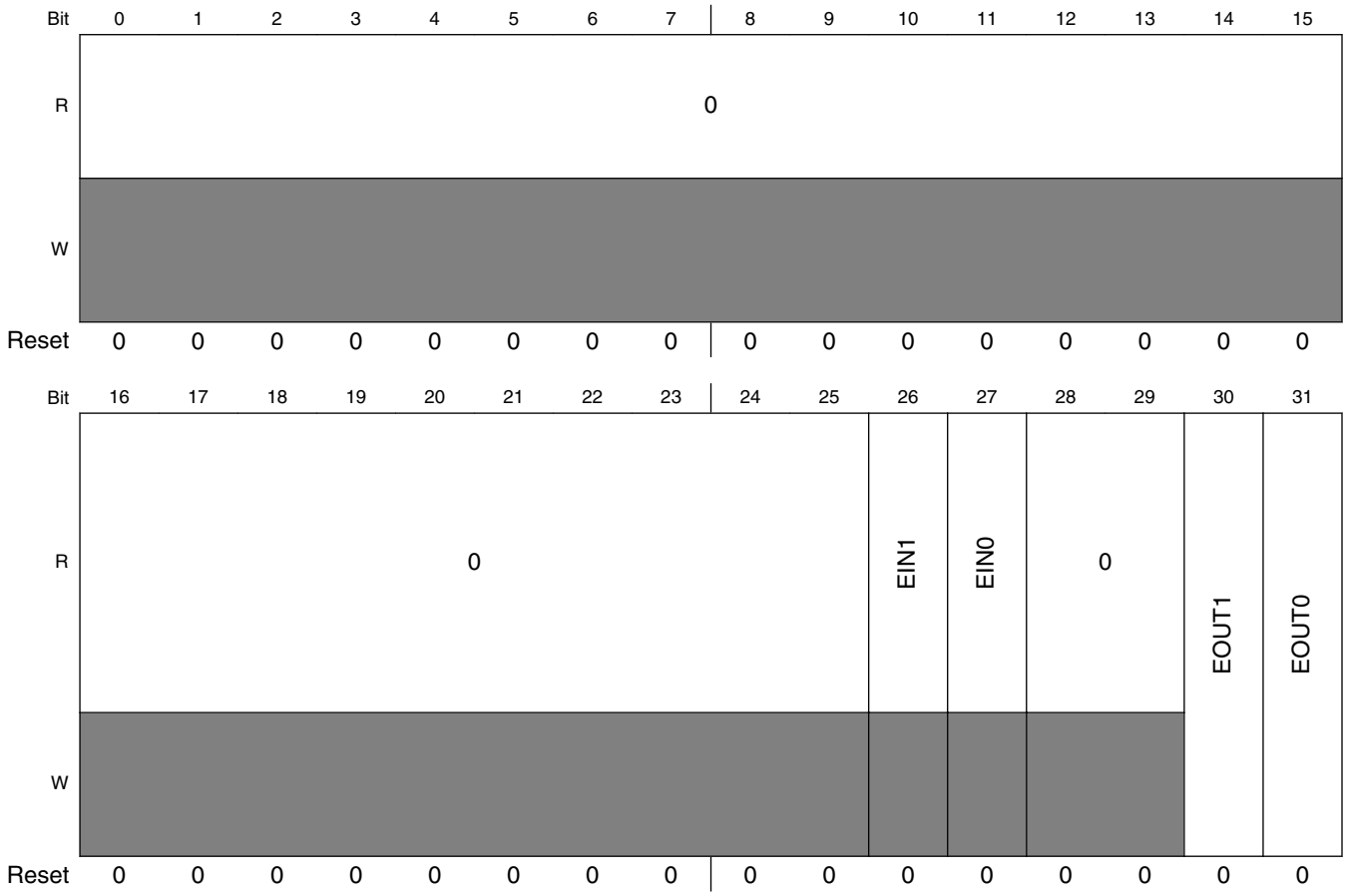
Table 85-12. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few IRCOSC clock cycles following a write/ read operation of the FCCU_EINOUT register.

Address: 0h base + BCh offset = BCh



FCCU_EINOUT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIN1	Error input 1. The EIN1 captures the respective fccu_ein[1] input signal. While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[1] pin. 0 when fccu_ein[1] = 0 1 when fccu_ein[1] = 1
27 EIN0	Error input 0.The EIN0 captures the fccu_ein[0] input signal. While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[0] pin. 0 when fccu_ein[0] = 0 1 when fccu_ein[0] = 1
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value. 0 force EOUT[1] = 0 1 force EOUT[1] = 1

Table continues on the next page...

FCCU_EINOUT field descriptions (continued)

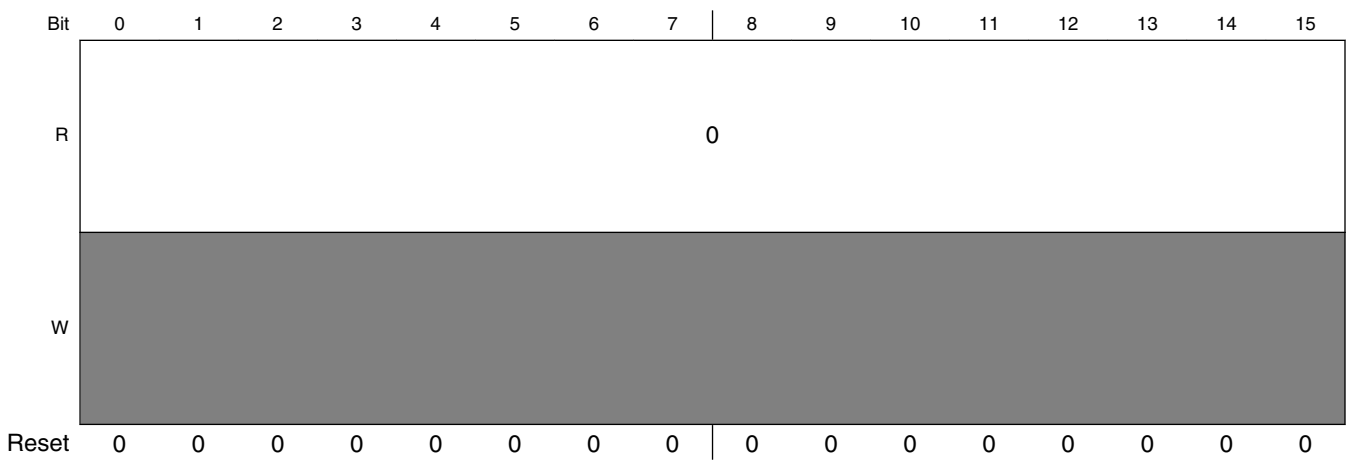
Field	Description
31 EOUT0	Error out 0 (significative only if the FCCU_CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value. 0 force EOUT[0] = 0 1 force EOUT[0] = 1

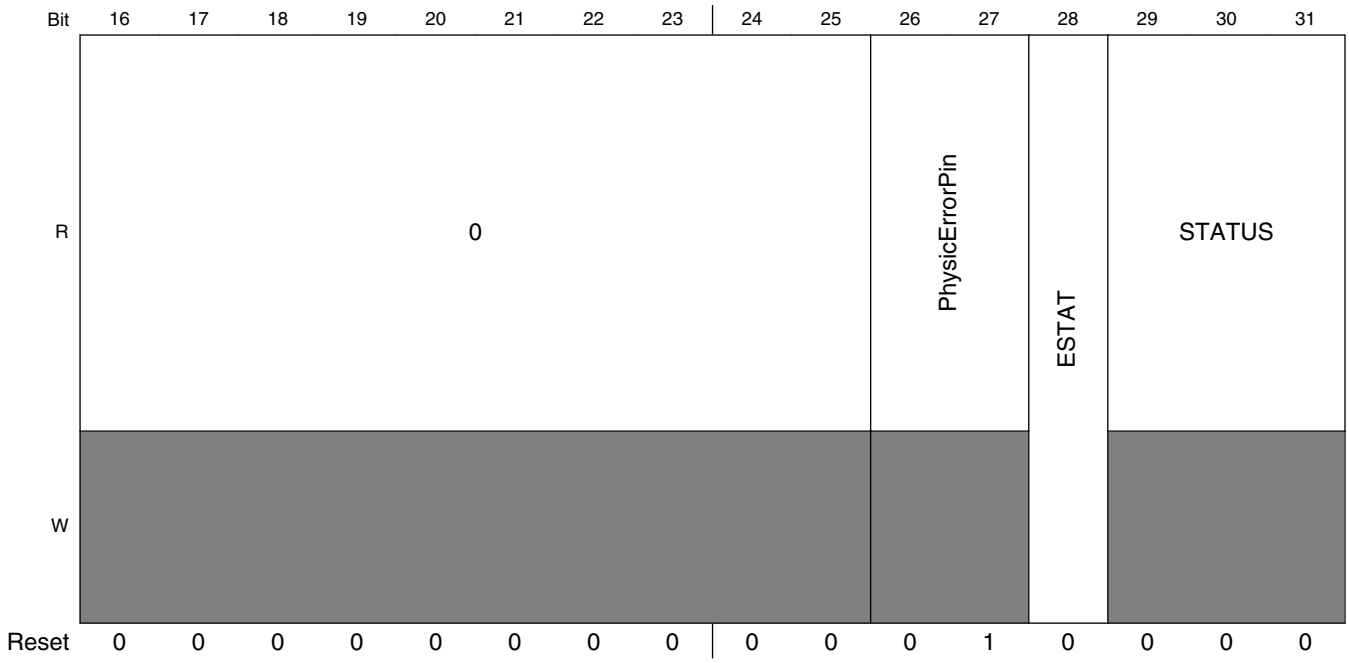
85.6.13 Status Register (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h





FCCU_STAT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	PhysicErrorPin : Physical Error Pin's state. PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0: Error Pin is at logical 0 1: Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol. 0 Error Pin is at logical 0 1 Error Pin is at logical 1
28 ESTAT	Current Error Pin Status. SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in non-faulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and SW can write it to 0 and after one cycle the bit reads 1 again. 0 System is OPERATIONAL (OK). 1 System is in FAULT state. This state is taken from the FCCU Eout logic (that is, as late as possible).
29–31 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state 100 Reserved

Table continues on the next page...

FCCU_STAT field descriptions (continued)

Field	Description
	101 Reserved 110 Reserved 111 Reserved

85.6.14 NA Freeze Status Register (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "recoverable" source (fccu_rf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

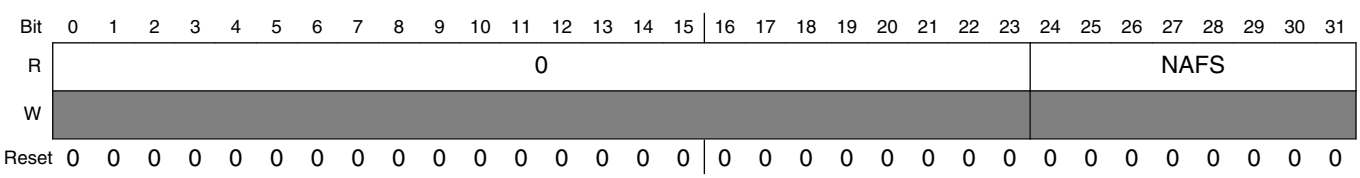
The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h



FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_rf[0] fault

Table continues on the next page...

FCCU_N2AF_STATUS field descriptions (continued)

Field	Description
	00000010 NORMAL to ALARM state transition cause => fccu_rf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_rf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_rf[127] fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_rf[] faults

85.6.15 AF Freeze Status Register (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the time-out trigger (recoverable fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

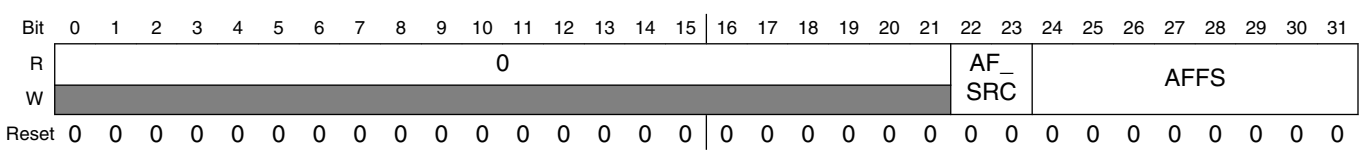
1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h



FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source **NOTE:** This field can be read and written but writing 1 has no effect.

Table continues on the next page...

FCCU_A2FF_STATUS field descriptions (continued)

Field	Description
	00 No fault 01 Reserved 10 Recoverable fault 11 Multiple and/or recoverable faults
24–31 AFFS	Alarm to Fault Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_rf[0] fault time-out fault 02h: ALARM to FAULT state transition cause => fccu_rf[1] fault time-out fault 03h: ALARM to FAULT state transition cause => fccu_rf[2] fault time-out fault ... 80h: ALARM to FAULT state transition cause => fccu_rf[127] fault time-out fault FFh: ALARM to FAULT state transition cause => multiple fccu_rf[] faults time-out faults

85.6.16 NF Freeze Status Register (FCCU_N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the source of the recoverable fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						NF_ SRC		NFFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source **NOTE:** This field can be read and written but writing 1 has no effect. 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple recoverable faults
24–31 NFFS	Normal to Fault Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_rf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_rf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_rf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_rf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_rf[] faults

85.6.17 FA Freeze Status Register (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the recoverable fault (fccu_rf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

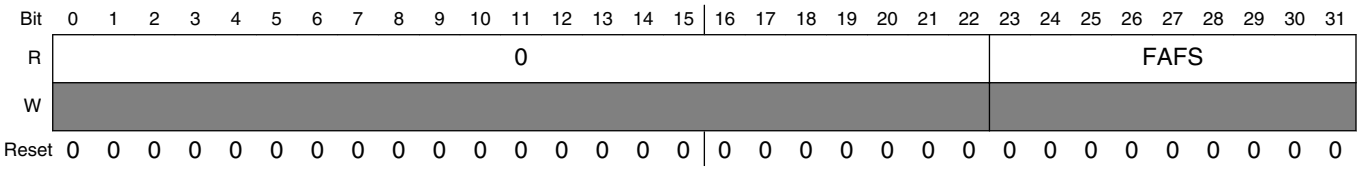
The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h



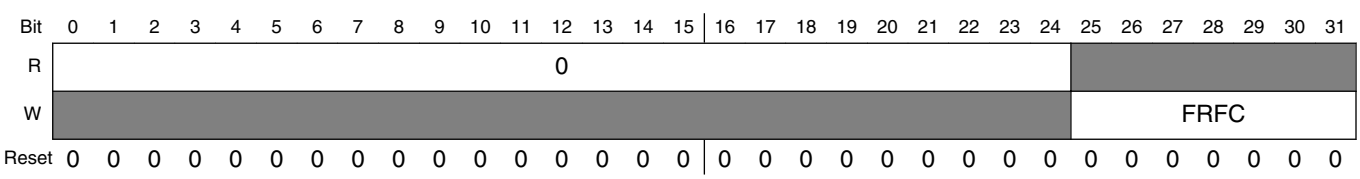
FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_rf[0] fault 02h: FAULT to ALARM state transition cause => fccu_rf[1] fault 03h: FAULT to ALARM state transition cause => fccu_rf[2] fault ... 80h: FAULT to ALARM state transition cause => fccu_rf[127] fault FFh: FAULT to ALARM state transition cause => multiple fccu_rf[] faults

85.6.18 RF Fake Register (FCCU_RFF)

The FCCU_RFF register contains a unique code to set a recovereable fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the recoverable faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake recoverable fault cannot be masked. The FCCU_RFF is a write-only register with a set of codes corresponding to each recoverable fault injection.

Address: 0h base + DCh offset = DCh



FCCU_RFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FRFC	FRFC : Fake recoverable fault code **NOTE:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake recoverablefault injection at recoverablefault source 0 01h: Fake recoverablefault injection at recoverablefault source 1 02h: Fake recoverablefault injection at recoverablefault source 2 ... 7Fh: Fake recoverablefault injection at recoverablefault source 127

85.6.19 IRQ Status Register (FCCU_IRQ_STAT)

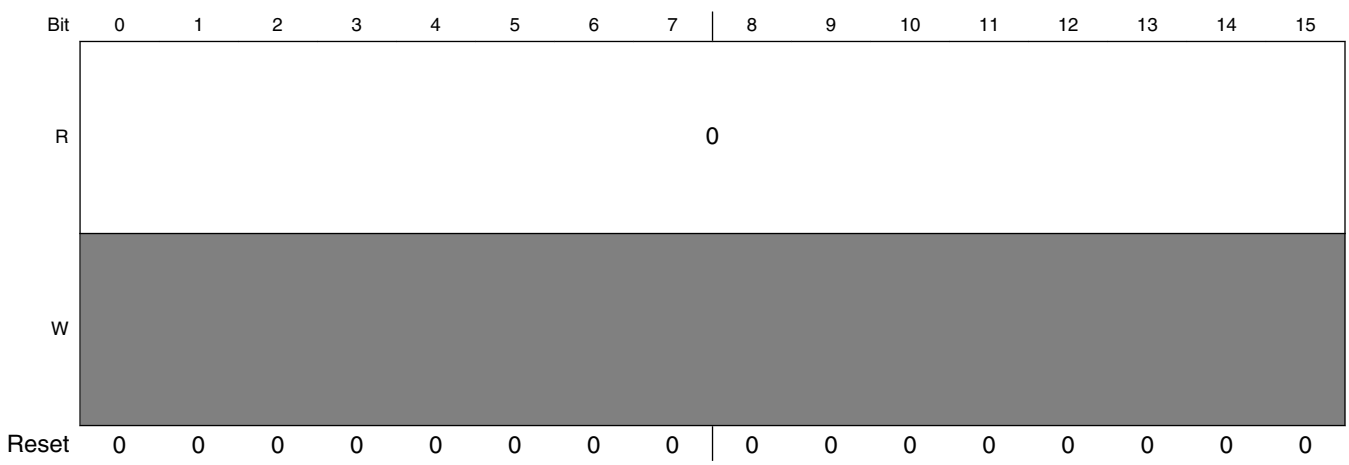
The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

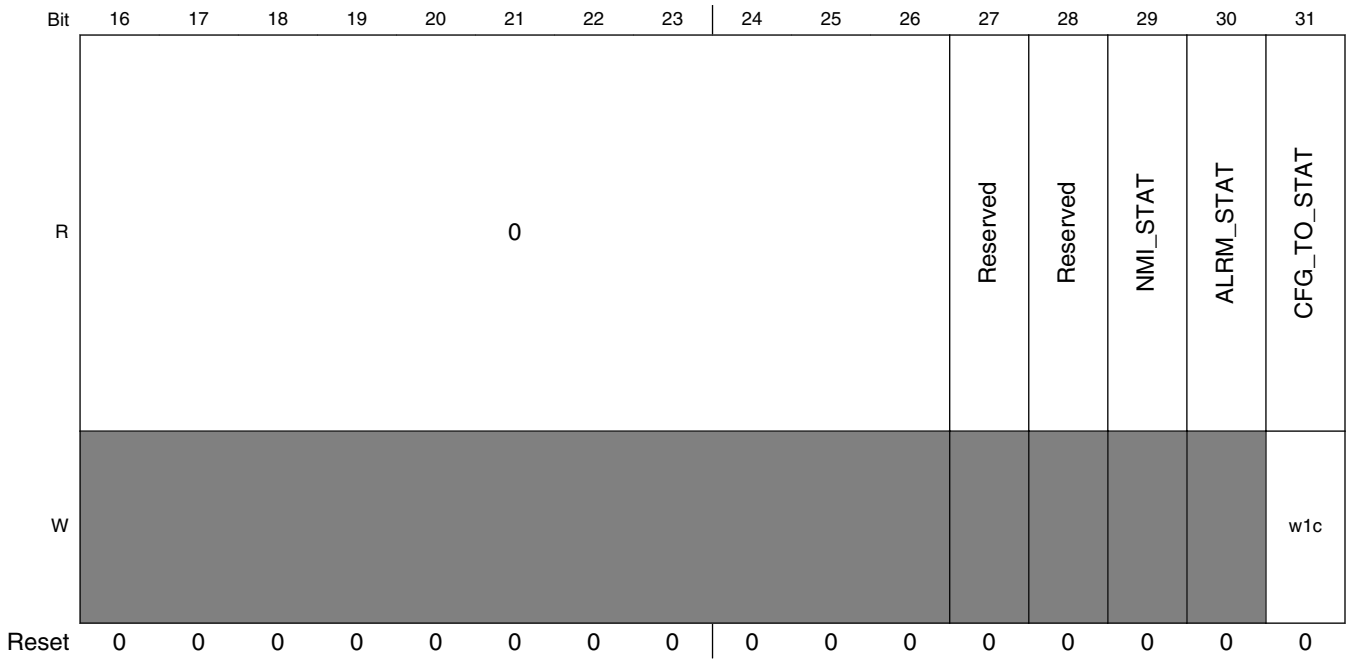
- Configuration time-out error
- Alarm interrupt
- NMI interrupt

The configuration time-out interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h





FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Time-out Status 0 No configuration time-out error 1 Configuration time-out error

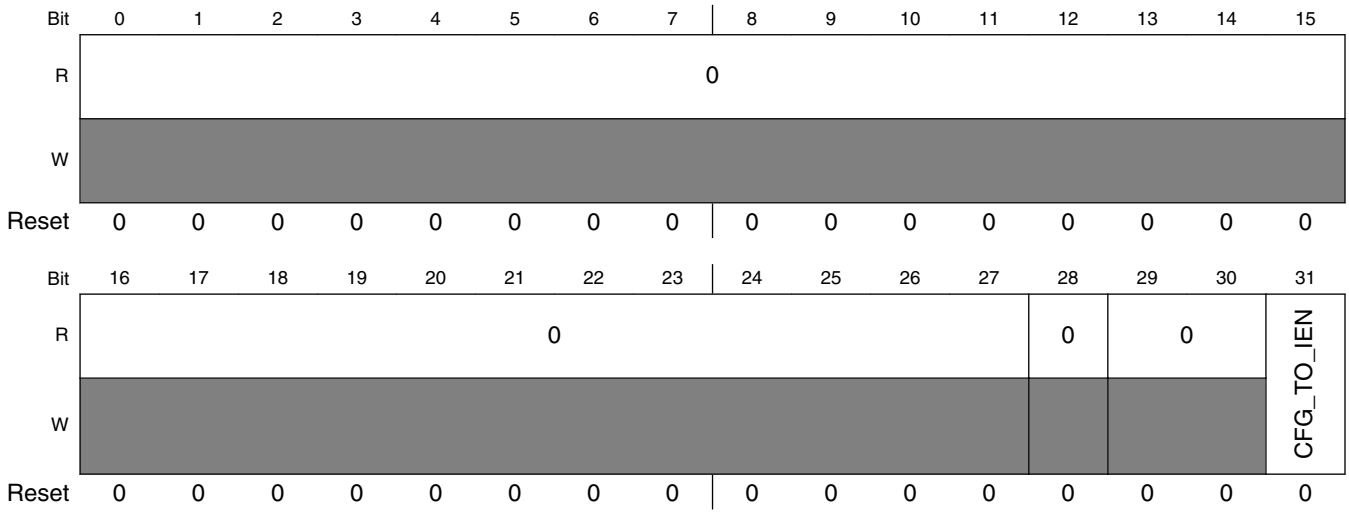
85.6.20 IRQ Enable Register (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration time-out error

The configuration time-out interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h



FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Time-out Interrupt Enable 0 Configuration time-out interrupt disabled 1 Configuration time-out interrupt enabled

85.6.21 XTMR Register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked on the IRCOSC clock.

[BLG_FCCU_0045][Covers: Saf3554]ETMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it goes high at the end of the T_min period from the last fault trigger.[end]

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   - OP17

- OP19
- OP20

2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 85-13 shows Timer state/value.

Table 85-13. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh
ETMR	00000000h	00000000h	00000000h	Running

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

85.6.22 MCS Register (FCCU_MCS)

NOTE

The reset value is chip-specific. See the chip-specific FCCU information.

The FCCU_MCS register contains a queue of the last four chip modes as defined by the Mode Entry module (MC_ME). MCS0 is the latest one, while MCS3 is the oldest one. In addition a qualifier indicates if the FCCU is in the FAULT state when the chip mode has been captured. The chip mode is provided by the MC_ME synchronous to the system clock whereas the FCCU captures the mode with the IRCOSC clock; therefore some uncertainty must be considered regarding the FAULT state indication.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0		MCS3				VL2	FS2	0		MCS2			
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0		MCS1				VL0	FS0	0		MCS0			
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- See the chip-specific FCCU information.x = Undefined at reset.

FCCU_MCS field descriptions

Field	Description
0 VL3	Valid It indicates that the correspondent MCS3 and FS3 fields are valid. 0 MCS3, FS3 fields are not significative 1 MCS3, FS3 fields are significative
1 FS3	Fault status It indicates that the correspondent MCS3 field has been captured when the FCCU is in FAULT state. 0 MCS3 field captured in any state different from the FAULT state 1 MCS3 field captured in FAULT state
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MCS3	MCS3: State of MC modules The MCS3 is the state of MC modules . MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual .
8 VL2	Valid It indicates that the correspondent MCS2 and FS2 fields are valid. 0 MCS2, FS2 fields are not significative 1 MCS2, FS2 fields are significative
9 FS2	Fault status It indicates that the correspondent MCS2 field has been captured when the FCCU is in FAULT state. 0 MCS2 field captured in any state different from the FAULT state 1 MCS2 field captured in FAULT state
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

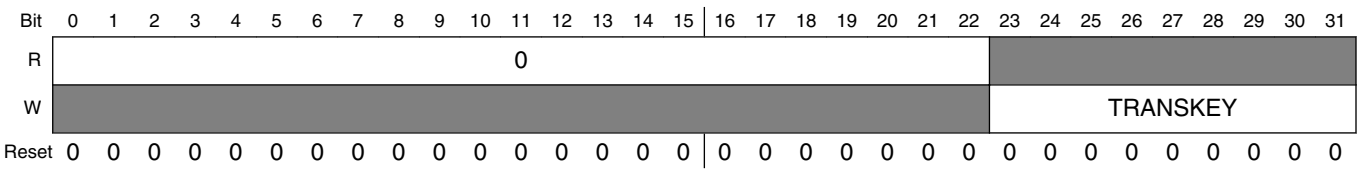
FCCU_MCS field descriptions (continued)

Field	Description
12–15 MCS2	MCS2: State of MC modules The MCS2 is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.
16 VL1	Valid It indicates that the correspondent MCS1 and FS1 fields are valid. 0 MCS1, FS1 fields are not significative 1 MCS1, FS1 fields are significative
17 FS1	Fault status It indicates that the correspondent MCS1 field has been captured when the FCCU is in FAULT state. 0 MCS1 field captured in any state different from the FAULT state 1 MCS1 field captured in FAULT state
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 MCS1	MCS1: State of MC modules The MCS1 is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.
24 VL0	Valid It indicates that the correspondent MCS0 and FS0 fields are valid. 0 MCS0, FS0 fields are not significative 1 MCS0, FS0 fields are significative
25 FS0	Fault status It indicates that the correspondent MCS0 field has been captured when the FCCU is in FAULT state. 0 MCS0 field captured in any state different from the FAULT state 1 MCS0 field captured in FAULT state
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MCS0	MCS0: State of MC modules The MCS0 is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.

85.6.23 Transient Lock Register (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing BCh. This register is available only in supervisor mode. This register is write only.

Address: 0h base + F0h offset = F0h



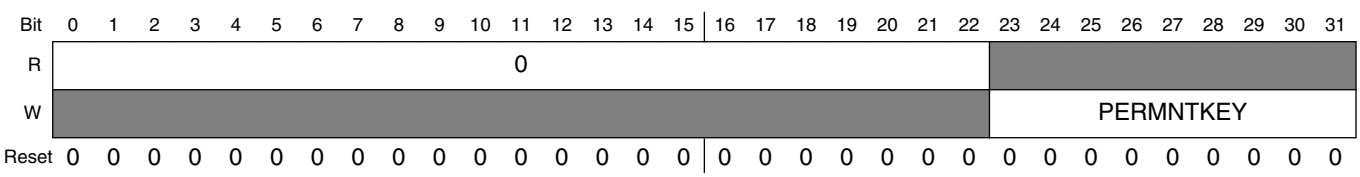
FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 TRANSKEY	Transition Unlocking Key value BCh: Configuration Unlocked Any other value:FCCU gets transiently locked again.

85.6.24 Permanent Lock Register (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Address: 0h base + F4h offset = F4h



FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FCCU_PERMNT_LOCK field descriptions (continued)

Field	Description
23–31 PERMNTKEY	Transition Locking Key value FFh: Configuration Locked Any other value: No Change

85.6.25 Delta T Register (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

NOTE

This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		Reserved													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 Reserved	This field is reserved.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	DELTA_T: Value of Delta_T in microseconds Max. value can be 16,666 microseconds (16.67 ms)

85.6.26 IRQ Alarm Enable Register (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

Table 85-14 shows FCCU IRQ alarm enable register channels.

Table 85-14. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0FCh	FCCU_IRQ_ALARM_EN0	IRQENE[31:0]
100h	FCCU_IRQ_ALARM_EN1	IRQENE[63:32]
104h	FCCU_IRQ_ALARM_EN2	IRQENE[95:64]
108h	FCCU_IRQ_ALARM_EN3	IRQENE[127:96]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable See Table 85-14 for register offset to channel number relationship. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

85.6.27 NMI Enable Register (FCCU_NMI_EN*n*)

These registers enable the NMI.

NOTE

Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Table 85-15 shows FCCU NMI enable register channels.

Table 85-15. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]
118h	FCCU_NMI_EN3	NMIENE[127:96]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NMI_EN*n* field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 85-15 for register offset to channel number relationship. 0 NMI is disabled for error (RF) source x. 1 NMI is enabled for error (RF) source x.

85.6.28 EOUT Signaling Enable Register (FCCU_EOUT_SIG_EN*n*)

These registers enable the error out signaling.

Table 85-16 shows FCCU EOUT signaling enable register channels.

Table 85-16. FCCU EOUT signaling enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
11Ch	FCCU_EOUT_SIG_EN0	EOUTENE[31:0]
120h	FCCU_EOUT_SIG_EN1	EOUTENE[63:32]
124h	FCCU_EOUT_SIG_EN2	EOUTENE[95:64]
128h	FCCU_EOUT_SIG_EN3	EOUTENE[127:96]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0–31 EOUTENx	EOUT signaling enable. See Table 85-16 for register offset to channel number relationship. 0 EOUT signaling is disabled for error (RF) source × for bistable protocol. Error pins are as if in non-faulty state. 1 EOUT signaling is enabled for error (RF) source × for bistable protocol.

85.7 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted, (i.e., an enabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by the IRCOSC. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FCCU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU, which is asserted on power-on, 'destructive', and external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

NOTE

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.

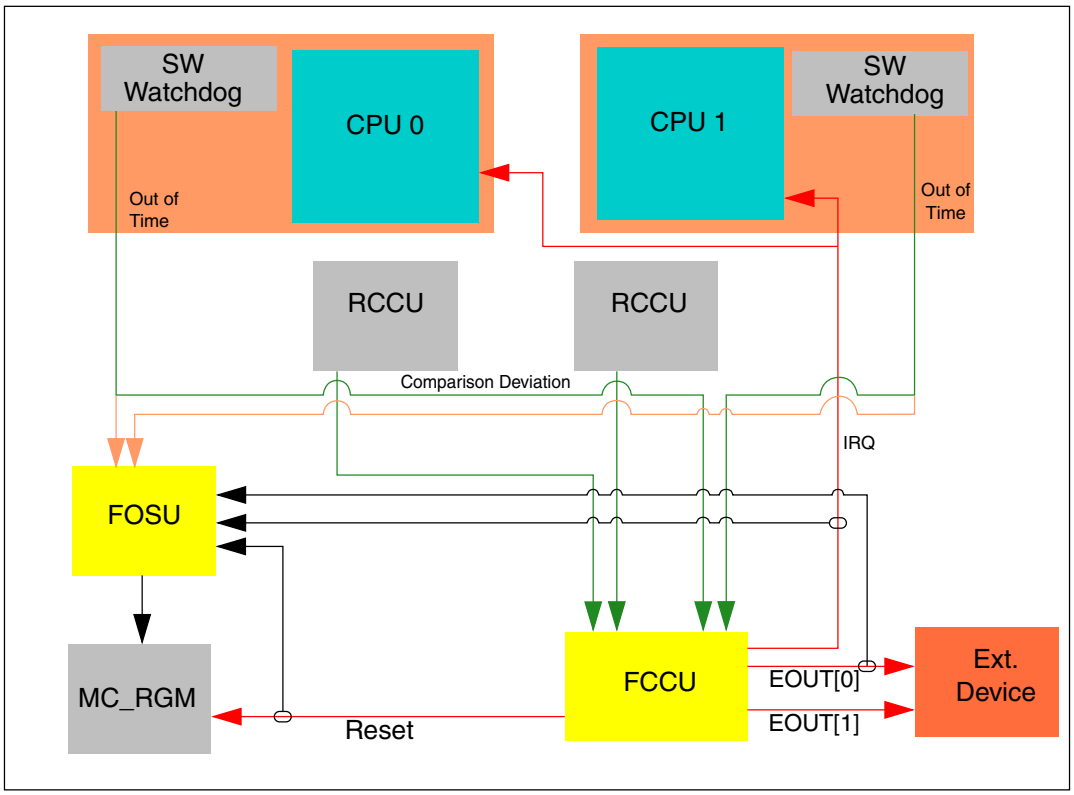


Figure 85-14. FOSU connections to the FCCU and MC_RGM

It is important to remember that there will be no FCCU reaction to a fault while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

NOTE

FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or destructive resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

85.8 Use cases and limitations

85.8.1 Misconfigurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enabling a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario

   Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.
4. If the user programs the FCCU for a higher timeout value than what is hardcoded for FOSU, then FCCU reacts later than what FOSU would be expecting and, as a result, FOSU will generate destructive reset request on its timeout.

85.8.2 Recommendations to configure FCCU

1. After a power on, 'destructive', or long external reset, where both system and FCCU are reset, the following steps could be followed to configure FCCU:
   a. Check and clear any pending fault status
   b. Verify FCCU is in NORMAL state, else repeat step(a) above
   c. Configure FCCU
2. After a short external or any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a. If active, wait for the Error out Tmin to expire
   b. Check and clear fault status
   c. Error pin moves to "non faulty" state, once fault status is cleared and Tmin expires
   d. Verify FCCU is in NORMAL state, else repeat step(a) above
   e. Read and verify value in FCCU_NCF_Ex
   f. Reconfigure FCCU, if necessary

Chapter 86
Self-Test Control Unit (STCU2)

86.1 Introduction

Self Test Control Unit (STCU2) is a comprehensive programmable hardware module that controls the self test sequence applied both during the Off-Line and/or On-Line conditions. It is able to manage by hardware the device's Logic Built-In Self Test (LBIST) and the SRAM/ROM Built-In Self Test (MBIST) blocks. The STCU2 includes the SSCM DCF bus to load the Self-test parameters (L/MBIST scheduling activity, LBIST setup, Unrecoverable/Recoverable faults management, CRC and MISR expected values, PLL management, and so on) from flash memory during the Off-Line Self-Test phase. This interface is able only to write the configuration parameters and start the Self-Test execution once after the STCU2 global reset has been applied. The register access by software is granted by an IPS interface with the purpose to check the results of the Off-Line Self-Test but also to load/check the execution of the On-Line Self-Test.

The STCU2 includes:

- A CRC block that can be optionally enabled to monitor the internal critical signals during the Self-Test run to increase the STCU2's Self-Checking capability
- A programmable Watchdog timer (WDG) to check the Self-Test operations (both LBIST and MBIST) have been completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before Watchdog time-out
- The PLL direct control during the Off-Line Self-Test sequence
- Interrupt lines enabled only during the On-Line Self-Test to simplify the software handling of Self-Test execution end
- Two sets of Result registers to collect the L/MBIST result during Off/On-Line Self-Test separately

To increase the security capability of this module, a different Security Key sequence is also required during Off/On-Line Self-Test in order to unlock the write access to the STCU2 registers. The STCU2 also implements two power-saving mechanisms to switch off:

- The global STCU2 clock after Self-Test sequence is completed
- The register ITF clock when the WDG time-out is detected after a Double Security Key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

86.2 Main features

- Double registers interface (SSCM DCF with priority over IPS)
- SSCM DCF write one time register interface
- IPS read and write (depends on SSCM enable parameters and STCU2_CFG.WRP bit) register interface
- Programmable scheduler for LBIST/MBIST execution
- LBIST concurrent/sequential execution
- MBIST concurrent/sequential execution
- Programmable LBIST delayed concurrent start
- Hardware capability to enable/disable the execution of hardware selected L/MBIST
- 32-bit CRC for STCU2 Self-Checking capability (enabled by STCU2_CFG.CRCEN bit)
- On-Line hardware Self-Test sequence abort capability
- On-Line software Self-Test sequence abort capability
- Programmable Internal clock prescaler to reduce Internal and MBIST/LBIST TCK clocks
- Programmable PLL direct control during Off-Line Self-test sequence
- PLL Lock signal monitoring during Off/On-Line Self Test
- L/MBIST dedicated On-Line interrupt source generation
- Programmable WDG Timer for Self-Check M/LBIST Time Execution
- Hard Coded WDG time-out to flag off-line initialization faults

- Hard Coded WDG time-out to Auto-lock STCU2 write access after Double-Key sequence
- LBIST and SRAM/ROM MBIST Off-Line status flags
- STCU2 internal errors Off-Line status flag
- LBIST and SRAM/ROM MBIST On-Line status flags
- STCU2 internal errors On-Line Status flag
- On-Line hardware abort status flag
- On-Line software abort status flag
- LBIST and SRAM/ROM MBIST Off/On-Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 internal errors Off-On/Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 errors (UF/RF) lines for FCCU signaling
- Redundant UF/RF generation logic to improve reliability
- FCCU UF/RF faults injection mechanism
- CRCE register access to perform redundant check by software
- LBIST MISRE registers access to perform redundant check by software
- Self-Test Bypass capability
- Global write register's protection mechanism based on Double Security Key
- STCU2 Global Auto Power-saving when Self-Test is completed, bypass mode is selected or WDT time-out is detected
- STCU2 ITF/WDG clock wake-up mechanism when software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)
- STCU2 ITF/WDG Auto Power-saving when Hard Coded WDG time-out is detected after software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)

86.3 Block diagram

The top level diagram of the STCU2 module is given in Figure 86-1.

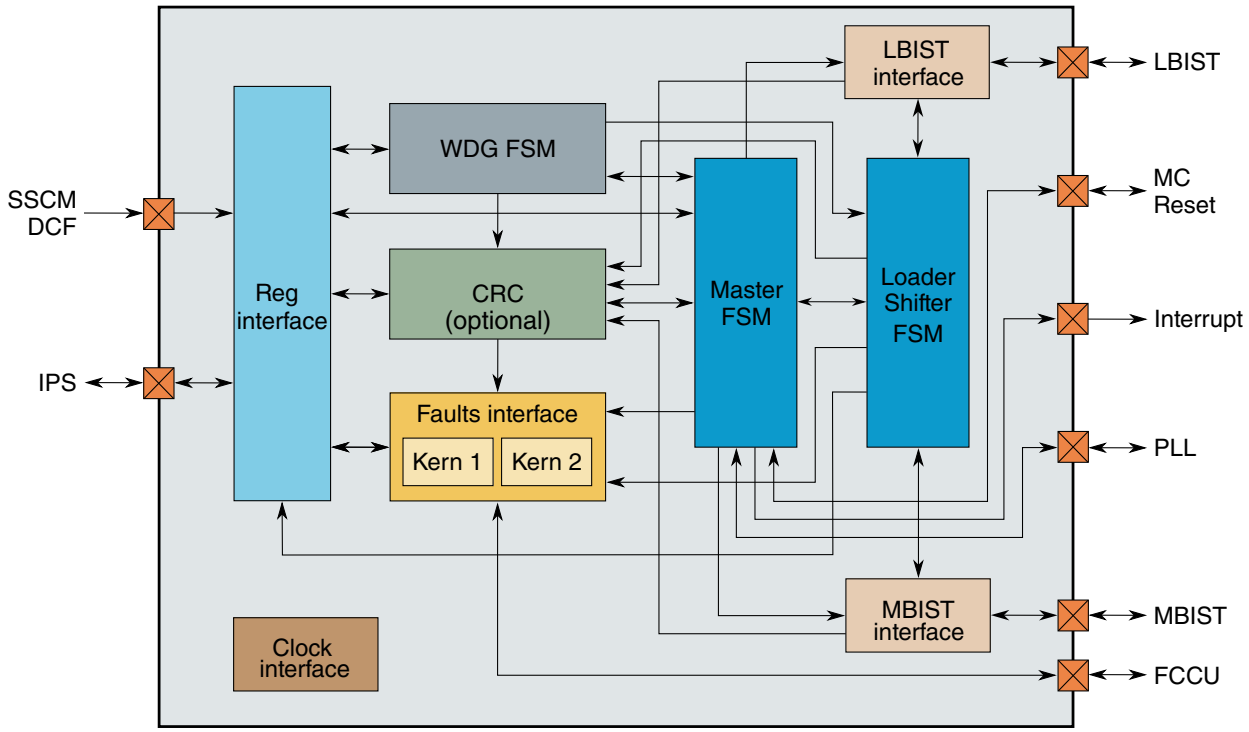


Figure 86-1. STCU2 block diagram

The STCU2 module includes the following sub-modules:

- ***Reg interface***: This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface
- ***CRC***: This module is the core of the Self-Checking architecture of the STCU2. It samples a set of selected internal signals when STCU2 is running.
- ***Fault interface***: This module collects the fault conditions related to L/MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each one of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the Set/Clear injection mechanism provided by the FCCU. To improve the intrinsic reliability of this critical logic, the generation logic is duplicated.
- ***Clock Interface***: This module manages the internal and the L/MBIST TCK clock prescaler, the internal Clock Gating Power Saving and the wake-up clock feature.
- ***WDG FSM***: This module includes two different Watchdogs. The first one is Hard Coded and is used to Auto Lock the STCU2 access forcing a reset condition on the Double Security Key registers while the second one has a double functionality. After a reset event initializes the STCU2, it is used as Hard Coded Watchdog time-out while during L/MBIST run it is used as programmable Watchdog timer to check the L/MBIST have been completed in the assigned L/MBIST time slot.

- ***STCU2 Master FSM***: This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the Self-Test Sequence.
- ***STCU2 Load Shifter FSM***: This module includes the common shifter register and the related state machine used to program the L/MBIST registers and read back the data to be checked at the end of each Test operation.
- ***LBIST interface***: This module includes the interface between the device's LBIST engines and the STCU2 controllers.
- ***MBIST interface***: This module includes the interface between the MBIST controller and STCU2 controllers.

86.4 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers
- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode only in the following scenario:

- The Self-Test sequence parameters are loaded before Off-Line Self-test is executed
- The IPS access on a specific register is allowed by software only when the bit STCU2_CFG.WRP is cleared
- In all the other conditions the write operations are not supported and all the STCU2 registers can be only read. Only exceptions are the following read/write bits: STCU2_CFG.WRP, STCU2_ERR_STAT.UFSF and STCU2_ERR_STAT.RFSF. See related register description for additional details

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral

- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied
- Any write operation performed on registers when the STCU2_CFG.WRP bit is set and the access is performed through software (IPS interface)
- Any write operation performed through software (IPS interface) on Read/On-Line_Write/Off-Line registers
- Any write operation performed off-line on registers in which writing operation is allowed only in the On-Line Self-Test phase when STCU2_CFG[WRP] = 1
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

NOTE

See the chip-specific STCU2 information for the wait time necessary to write to the online registers.

86.5 Functional description

86.5.1 FSM description

The module has three state machines that work together: Master State Machine, Loader/Shifter State Machine, and the Watchdog State Machine. Basically the Master State Machine is the core unit of the STCU2 module. It coordinates all the Self Test operations and the other State Machines. The Loader/Shifter State Machine is used to program the MBIST and the LBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master State Machine. The Watchdog State Machine evaluates all the schedule time for MBIST and LBIST and the time-out in case of wrong STCU2 programming.

86.5.2 Reset management

Assertion of the STCU2 reset input signal initializes the STCU2 module status, forcing the Off-Line Self-Test condition and cleaning up all the registers and state machine of the module.

The reset signals generated by the STCU2 module at the end of the Self-Test execution phase depends on the Self-Test conditions and on the content of the LBRMSW register.

86.5.2.1 Off-Line reset generation

At the end of the Off-Line Self-Test execution the STCU2 raises the global functional reset.

86.5.2.2 On-Line reset generation (only MBIST)

The On-Line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences related to the write operations performed by the MBIST execution other than preventing any read/write operation during the Self-Test execution. If necessary, the software must also restore the memory content (only for RAMs) at the end of the Self-Test execution.

86.5.2.3 On-Line reset generation (LBIST enabled)

The LBRMSW register should be configured to generate a global functional reset after LBIST online self-test. It is not recommended to use a dedicated reset due to random data injected by the LBIST Controller.

Note

In case there is an ABORT condition detected by STCU2 during the On-Line Self-Test execution, independently of the value of the LBRMSW register, only the related reset line stcu2_rst_lbist_core_b[NLBIST-1:0] is pulsed to prevent any potential dangerous system level operations.

86.5.3 Built-in self-test scheduling

The STCU2 module has been designed to be very flexible allowing to program the parallel/serial execution of the MBIST or LBIST depending on the power/timing/ coverage constraints. The limitation in the programming flexibility is described in Design implementation information.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU2_CFG.PTR. The first LBIST is mapped on 00h, the second on 01h, and so on. The first MBIST is mapped at the address 10h, the second on 11h and so on. The additional pointers are in the bitfield STCU2_LB_CTRL[X].PTR for LBIST[X] (where X is the selected LBIST) and STCU2_MB_CTRL[Y].PTR for MBIST (where Y is the selected MBIST) and have to be filled depending on the selected sequence of run. The additional bitfield STCU2_LB_CTRL.CSM and STCU2_MB_CTRL.CSM provide the flexibility to run concurrently or sequentially the chosen set of the LBIST or MBIST or to close the linked list setting the NIL pointer. For more details, see STCU2 LBIST Control Register (STCU2_LB_CTRL*n*) and STCU2 MBIST Control Register (STCU2_MB_CTRL*n*).

NOTE

Program operation should ensure that no flash write/erase operations are ongoing when online LBIST is triggered. Otherwise, a flash segment may be corrupted.

86.5.4 ABORT management

The STCU2 module provides two mechanisms to perform the On-Line Self-Test execution abort. When the abort is detected in both cases, the On-Line Self-Test operation stops running and the status of the currently running MBIST or LBIST is saved into the related registers to provide intermediate results that might be useful in case of debug or in case of concurrent run. In case of LBIST run, see Reset management, for more details about reset specific management when abort is detected.The following sections describe the differences between these two modalities.

86.5.4.1 Hardware ABORT management

The STCU2 enters the hardware abort condition as a consequence of a Functional reset which causes Self-Test execution to abort .

When hardware abort is detected the bit ABORTHW of the STCU2_ERR_STAT register is set. This flag allows the software to diagnose what has happened during the On-Line Self-Test execution run.

86.5.4.2 Software ABORT management

The STCU2 enters in the abort condition because the software set the bit RUNSW_ABORT in the STCU2_RUNSW register. In order to allow the software to understand when STCU2 has completed the abort sequence, this bit will be maintained active until the abort sequence is completed. At that time the STCU2 will clear this bit as described in STCU2 Run Software Register (STCU2_RUNSW).

When software abort is raised the bit ABORTSW of the STCU2_ERR_STAT register is set. Even if the abort condition is forced by software operation, this flag allows to diagnose what has happened during the On-Line Self-Test execution run.

86.5.5 PLL interface

The STCU2 module is able to directly control the PLL during the Off-Line Self-Test operations depending on the status of the STCU2_RUN[MBPLLEN] and STCU2_RUN[LBPLLEN].

When STCU2_RUN[RUN] is set to start the Off-Line Self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed into the STCU2_PLL_CFG register.

When the STCU2_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock signal. As soon as the lock happens, the STCU2 forces the clock to switch to the PLL output clock. This allows the MBIST to proceed at the PLL output frequency. At the end of the MBIST run the clock source is reverted to the original one.

Same behavior will be applied when STCU2_RUN[LBPLLEN] is active and LBIST is currently selected.

86.5.6 Interrupt interface

Interrupt sources:

- Triggered by MBIST (enabled by programming STCU2_RUNSW[MBIE] = 1)
- Triggered by LBIST (enabled by programming STCU2_RUNSW[LBIE] = 1)

As described in the STCU2 Run Software Register (STCU2_RUNSW) description, the interrupts are generated at the end of any concurrent run. This means that in case of multiple sequential run in the same session, the number of interrupts generated is equivalent to the number of the sequential session providing in such a way a fine granularity with respect to the completed run.

The LBIFLG bit and MBIFLG bit in the STCU2 Interrupt Flag Register (STCU2_INT_FLG) register flag interrupt conditions and allow the clearing of interrupt requests. Interrupt bits are cleared by writing the related flag bit to 1 to enable the next interrupt request.

86.5.7 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable fault and/or a Recoverable fault failure during the Self Test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

86.5.8 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished in time.

86.5.8.1 Initialization watchdog timer

The STCU2 has two initialization operating modes: program the self-test sequence and run it (Safety mode) or bypass completely the self-test sequence setting the bit STCU2_CFG.BYP. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded watchdog time-out flags the wrong behavior.

86.5.8.2 Built-in self-test watchdog timer

The LBIST and MBIST execution time has to be configured as described in STCU2 Watchdog Register Granularity (STCU2_WDG), to account for the overall execution time of the Self-Test sequence. In case the selected LBISTs or MBISTs are not yet completed during the Off-line Self-Test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into STCU2_ERR_STAT.WDTO and STCU2_MBEL/M/H or STCU2_LBE registers while in case of On-Line Self-Test into STCU2_ERR_STAT.WDTOSW and STCU2_MBELSW/MSW/HSW or STCU2_LBESW.

In case of multiple sequential run in the same On-Line/Off-Line session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

In all the cases, the STCU2 reset generation is managed in one of the modes described in Reset management.

86.5.8.3 Register write-access watchdog timer

The access on the STCU2 registers during the self-test configuration phase (both online/offline) is protected by a key mechanism to prevent any unwanted access, as described in the STCU2_SKC register description. In addition to this malicious access protection feature, there is a hardware watchdog timer that, after AUTOLOCK_VALUE STCU2 clock cycles, lock the access requiring to apply again the STCU2_SKC keys.

NOTE

Each DCF record takes 8 STCU2 clock cycles for write completion. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is AUTOLOCK Value divided by 8. To refresh the hardware watchdog timer, the user must write key 2 into the STCU2_SKC register.

This feature is particularly useful in case STCU2_CFG[WRP] is 0 during the software self-test configuration since in this case the software application might enable the write access to the STCU2 registers.

86.5.9 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomial to evaluate the signature of the most important and critical internal signals. This feature is by default switched off but it can be turned on setting the bit STCU2_CFG.CRCEN. Once this bit is set the expected signature has to be programmed into the STCU2_CRCE register; at the end of the Self-Test execution the STCU2 updates the content of the STCU2_CRCR register with the evaluated signature and compares this value with the expected one previously programmed into STCU2_CRCE. In case of mismatches a UF/RF condition is flagged depending on the fault mapping selected into the STCU2_ERR_FM register.

NOTE

The CRC calculation depends on the internal state of the STCU2 when the online/offline self test is started. The STCU2's internal state is dependent on DCF values and their programming sequence for offline selftest. The calculated CRC value can be determined by running an LBIST test under known conditions. The CRC unit produces a stable and predictable result but is dependent on several stimuli (including programming order and the initial internal state of the STCU2 design) during online self test. To calculate a CRC value, run an LBIST test on known good silicon with a predefined software programming sequence. Record the CRC result value. Then provide the STCU2 programming sequence and the initial register vlaues for further runs if the CRC feature is to be used. Once a known good CRC value is determined, the CRC value will not change from one run to another so long as the user does not change the STCU2 programming sequence or any of the values used to program STCU2 registers. To obtain a consistant CRC calculated value, the STCU must start from the same state compared to the state when the CRC was originally evaluated.

86.5.10 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without the CPU intervention after a reset trigger event initializes the STCU2. This bus interface has the priority over the IPS.

86.5.11 L/MBIST execution restriction

The STCU2 provide the capability of preventing the execution of selected L/MBIST. The list of the L/MBIST with this restriction is Hardware defined at design phase STCU2_LBIST_SYS_ENSTCU2_MBIST_SYS_EN) and cannot be changed by software. The activation of the restriction is controlled by system signal sys_lmbist_restriction_en which has to be carefully generated in order to fulfill the system requirements.

86.6 Register description

During Offline SELFTEST, once the STCU2 registers are configured they cannot be overridden.

During Online SELFTEST, once the STCU2 registers are configured they cannot be overridden via IPS till the selftest is complete.

The STCU2 registers are listed in this section.

STCU2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STCU2 Run Register (STCU2_RUN)	32	R/W	0000_0000h	86.6.1/4475
4	STCU2 Run Software Register (STCU2_RUNSW)	32	R/W	0000_0000h	86.6.2/4476
8	STCU2 SK Code Register (STCU2_SKC)	32	W	0000_0000h	86.6.3/4478
C	STCU2 Configuration Register (STCU2_CFG)	32	R/W	See section	86.6.4/4479
10	STCU2 PLL Configuration Register (STCU2_PLL_CFG)	32	R/W	See section	86.6.5/4482
14	STCU2 Watchdog Register Granularity (STCU2_WDG)	32	R/W	See section	86.6.6/4483
18	STCU2 Interrupt Flag Register (STCU2_INT_FLG)	32	w1c	0000_0000h	86.6.7/4485
1C	STCU2 CRC Expected Status Register (STCU2_CRCE)	32	R/W	Undefined	86.6.8/4486
20	STCU2 CRC Read Status Register (STCU2_CRCR)	32	R	See section	86.6.9/4486
24	STCU2 Error Register (STCU2_ERR_STAT)	32	R/W	0000_0000h	86.6.10/ 4487
28	STCU2 Error FM Register (STCU2_ERR_FM)	32	R/W	See section	86.6.11/ 4491
2C	STCU2 Off-Line LBIST Status Register (STCU2_LBS)	32	R	0000_0000h	86.6.12/ 4492
30	STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)	32	R	0000_0000h	86.6.13/ 4493
34	STCU2 On-Line LBIST Status Register (STCU2_LBSSW)	32	R	0000_0000h	86.6.14/ 4495

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38	STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)	32	R	0000_0000h	86.6.15/ 4496
3C	STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)	32	R/W	See section	86.6.16/ 4498
40	STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	32	R/W	See section	86.6.17/ 4500
44	STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)	32	R	0000_0000h	86.6.18/ 4502
48	STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)	32	R	0000_0000h	86.6.19/ 4507
4C	STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)	32	R	0000_0000h	86.6.20/ 4512
50	STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)	32	R	0000_0000h	86.6.21/ 4515
54	STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)	32	R	0000_0000h	86.6.22/ 4519
58	STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)	32	R	0000_0000h	86.6.23/ 4522
5C	STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)	32	R	0000_0000h	86.6.24/ 4524
60	STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)	32	R	0000_0000h	86.6.25/ 4528
64	STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)	32	R	0000_0000h	86.6.26/ 4532
68	STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)	32	R	0000_0000h	86.6.27/ 4534
6C	STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)	32	R	0000_0000h	86.6.28/ 4538
70	STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)	32	R	0000_0000h	86.6.29/ 4541
74	STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	32	R/W	See section	86.6.30/ 4543
78	STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	32	R/W	See section	86.6.31/ 4547
7C	STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	32	R/W	See section	86.6.32/ 4551
100	STCU2 LBIST Control Register (STCU2_LB_CTRL0)	32	R/W	See section	86.6.33/ 4553
104	STCU2 LBIST PC Stop Register (STCU2_LB_PCS0)	32	R/W	See section	86.6.34/ 4556
108	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL0)	32	R/W	See section	86.6.35/ 4557
10C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH0)	32	R/W	See section	86.6.36/ 4557

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
110	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL0)	32	R/W	See section	86.6.37/ 4558
114	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH0)	32	R/W	See section	86.6.38/ 4559
118	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL0)	32	R	See section	86.6.39/ 4559
11C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH0)	32	R	See section	86.6.40/ 4560
120	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW0)	32	R/W	See section	86.6.41/ 4561
124	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW0)	32	R/W	See section	86.6.42/ 4561
128	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW0)	32	R	See section	86.6.43/ 4562
12C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW0)	32	R	See section	86.6.44/ 4563
140	STCU2 LBIST Control Register (STCU2_LB_CTRL1)	32	R/W	See section	86.6.33/ 4553
144	STCU2 LBIST PC Stop Register (STCU2_LB_PCS1)	32	R/W	See section	86.6.34/ 4556
148	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL1)	32	R/W	See section	86.6.35/ 4557
14C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH1)	32	R/W	See section	86.6.36/ 4557
150	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL1)	32	R/W	See section	86.6.37/ 4558
154	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH1)	32	R/W	See section	86.6.38/ 4559
158	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL1)	32	R	See section	86.6.39/ 4559
15C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH1)	32	R	See section	86.6.40/ 4560
160	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW1)	32	R/W	See section	86.6.41/ 4561
164	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW1)	32	R/W	See section	86.6.42/ 4561
168	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW1)	32	R	See section	86.6.43/ 4562
16C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW1)	32	R	See section	86.6.44/ 4563
180	STCU2 LBIST Control Register (STCU2_LB_CTRL2)	32	R/W	See section	86.6.33/ 4553
184	STCU2 LBIST PC Stop Register (STCU2_LB_PCS2)	32	R/W	See section	86.6.34/ 4556

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
188	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL2)	32	R/W	See section	86.6.35/ 4557
18C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH2)	32	R/W	See section	86.6.36/ 4557
190	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL2)	32	R/W	See section	86.6.37/ 4558
194	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH2)	32	R/W	See section	86.6.38/ 4559
198	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL2)	32	R	See section	86.6.39/ 4559
19C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH2)	32	R	See section	86.6.40/ 4560
1A0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW2)	32	R/W	See section	86.6.41/ 4561
1A4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW2)	32	R/W	See section	86.6.42/ 4561
1A8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW2)	32	R	See section	86.6.43/ 4562
1AC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW2)	32	R	See section	86.6.44/ 4563
1C0	STCU2 LBIST Control Register (STCU2_LB_CTRL3)	32	R/W	See section	86.6.33/ 4553
1C4	STCU2 LBIST PC Stop Register (STCU2_LB_PCS3)	32	R/W	See section	86.6.34/ 4556
1C8	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL3)	32	R/W	See section	86.6.35/ 4557
1CC	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH3)	32	R/W	See section	86.6.36/ 4557
1D0	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL3)	32	R/W	See section	86.6.37/ 4558
1D4	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH3)	32	R/W	See section	86.6.38/ 4559
1D8	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL3)	32	R	See section	86.6.39/ 4559
1DC	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH3)	32	R	See section	86.6.40/ 4560
1E0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW3)	32	R/W	See section	86.6.41/ 4561
1E4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW3)	32	R/W	See section	86.6.42/ 4561
1E8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW3)	32	R	See section	86.6.43/ 4562
1EC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW3)	32	R	See section	86.6.44/ 4563

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200	STCU2 LBIST Control Register (STCU2_LB_CTRL4)	32	R/W	See section	86.6.33/ 4553
204	STCU2 LBIST PC Stop Register (STCU2_LB_PCS4)	32	R/W	See section	86.6.34/ 4556
208	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL4)	32	R/W	See section	86.6.35/ 4557
20C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH4)	32	R/W	See section	86.6.36/ 4557
210	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL4)	32	R/W	See section	86.6.37/ 4558
214	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH4)	32	R/W	See section	86.6.38/ 4559
218	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL4)	32	R	See section	86.6.39/ 4559
21C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH4)	32	R	See section	86.6.40/ 4560
220	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW4)	32	R/W	See section	86.6.41/ 4561
224	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW4)	32	R/W	See section	86.6.42/ 4561
228	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW4)	32	R	See section	86.6.43/ 4562
22C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW4)	32	R	See section	86.6.44/ 4563
240	STCU2 LBIST Control Register (STCU2_LB_CTRL5)	32	R/W	See section	86.6.33/ 4553
244	STCU2 LBIST PC Stop Register (STCU2_LB_PCS5)	32	R/W	See section	86.6.34/ 4556
248	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL5)	32	R/W	See section	86.6.35/ 4557
24C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH5)	32	R/W	See section	86.6.36/ 4557
250	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL5)	32	R/W	See section	86.6.37/ 4558
254	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH5)	32	R/W	See section	86.6.38/ 4559
258	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL5)	32	R	See section	86.6.39/ 4559
25C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH5)	32	R	See section	86.6.40/ 4560
260	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW5)	32	R/W	See section	86.6.41/ 4561
264	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW5)	32	R/W	See section	86.6.42/ 4561

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
268	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW5)	32	R	See section	86.6.43/ 4562
26C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW5)	32	R	See section	86.6.44/ 4563
280	STCU2 LBIST Control Register (STCU2_LB_CTRL6)	32	R/W	See section	86.6.33/ 4553
284	STCU2 LBIST PC Stop Register (STCU2_LB_PCS6)	32	R/W	See section	86.6.34/ 4556
288	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL6)	32	R/W	See section	86.6.35/ 4557
28C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH6)	32	R/W	See section	86.6.36/ 4557
290	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL6)	32	R/W	See section	86.6.37/ 4558
294	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH6)	32	R/W	See section	86.6.38/ 4559
298	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL6)	32	R	See section	86.6.39/ 4559
29C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH6)	32	R	See section	86.6.40/ 4560
2A0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW6)	32	R/W	See section	86.6.41/ 4561
2A4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW6)	32	R/W	See section	86.6.42/ 4561
2A8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW6)	32	R	See section	86.6.43/ 4562
2AC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW6)	32	R	See section	86.6.44/ 4563
2C0	STCU2 LBIST Control Register (STCU2_LB_CTRL7)	32	R/W	See section	86.6.33/ 4553
2C4	STCU2 LBIST PC Stop Register (STCU2_LB_PCS7)	32	R/W	See section	86.6.34/ 4556
2C8	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL7)	32	R/W	See section	86.6.35/ 4557
2CC	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH7)	32	R/W	See section	86.6.36/ 4557
2D0	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL7)	32	R/W	See section	86.6.37/ 4558
2D4	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH7)	32	R/W	See section	86.6.38/ 4559
2D8	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL7)	32	R	See section	86.6.39/ 4559
2DC	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH7)	32	R	See section	86.6.40/ 4560

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2E0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW7)	32	R/W	See section	86.6.41/ 4561
2E4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW7)	32	R/W	See section	86.6.42/ 4561
2E8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW7)	32	R	See section	86.6.43/ 4562
2EC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW7)	32	R	See section	86.6.44/ 4563
300	STCU2 LBIST Control Register (STCU2_LB_CTRL8)	32	R/W	See section	86.6.33/ 4553
304	STCU2 LBIST PC Stop Register (STCU2_LB_PCS8)	32	R/W	See section	86.6.34/ 4556
308	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL8)	32	R/W	See section	86.6.35/ 4557
30C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH8)	32	R/W	See section	86.6.36/ 4557
310	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL8)	32	R/W	See section	86.6.37/ 4558
314	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH8)	32	R/W	See section	86.6.38/ 4559
318	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL8)	32	R	See section	86.6.39/ 4559
31C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH8)	32	R	See section	86.6.40/ 4560
320	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW8)	32	R/W	See section	86.6.41/ 4561
324	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW8)	32	R/W	See section	86.6.42/ 4561
328	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW8)	32	R	See section	86.6.43/ 4562
32C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW8)	32	R	See section	86.6.44/ 4563
340	STCU2 LBIST Control Register (STCU2_LB_CTRL9)	32	R/W	See section	86.6.33/ 4553
344	STCU2 LBIST PC Stop Register (STCU2_LB_PCS9)	32	R/W	See section	86.6.34/ 4556
348	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL9)	32	R/W	See section	86.6.35/ 4557
34C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH9)	32	R/W	See section	86.6.36/ 4557
350	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL9)	32	R/W	See section	86.6.37/ 4558
354	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH9)	32	R/W	See section	86.6.38/ 4559

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
358	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL9)	32	R	See section	86.6.39/ 4559
35C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH9)	32	R	See section	86.6.40/ 4560
360	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW9)	32	R/W	See section	86.6.41/ 4561
364	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW9)	32	R/W	See section	86.6.42/ 4561
368	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW9)	32	R	See section	86.6.43/ 4562
36C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW9)	32	R	See section	86.6.44/ 4563
600	STCU2 MBIST Control Register (STCU2_MB_CTRL0)	32	R/W	See section	86.6.45/ 4564
604	STCU2 MBIST Control Register (STCU2_MB_CTRL1)	32	R/W	See section	86.6.45/ 4564
608	STCU2 MBIST Control Register (STCU2_MB_CTRL2)	32	R/W	See section	86.6.45/ 4564
60C	STCU2 MBIST Control Register (STCU2_MB_CTRL3)	32	R/W	See section	86.6.45/ 4564
610	STCU2 MBIST Control Register (STCU2_MB_CTRL4)	32	R/W	See section	86.6.45/ 4564
614	STCU2 MBIST Control Register (STCU2_MB_CTRL5)	32	R/W	See section	86.6.45/ 4564
618	STCU2 MBIST Control Register (STCU2_MB_CTRL6)	32	R/W	See section	86.6.45/ 4564
61C	STCU2 MBIST Control Register (STCU2_MB_CTRL7)	32	R/W	See section	86.6.45/ 4564
620	STCU2 MBIST Control Register (STCU2_MB_CTRL8)	32	R/W	See section	86.6.45/ 4564
624	STCU2 MBIST Control Register (STCU2_MB_CTRL9)	32	R/W	See section	86.6.45/ 4564
628	STCU2 MBIST Control Register (STCU2_MB_CTRL10)	32	R/W	See section	86.6.45/ 4564
62C	STCU2 MBIST Control Register (STCU2_MB_CTRL11)	32	R/W	See section	86.6.45/ 4564
630	STCU2 MBIST Control Register (STCU2_MB_CTRL12)	32	R/W	See section	86.6.45/ 4564
634	STCU2 MBIST Control Register (STCU2_MB_CTRL13)	32	R/W	See section	86.6.45/ 4564
638	STCU2 MBIST Control Register (STCU2_MB_CTRL14)	32	R/W	See section	86.6.45/ 4564
63C	STCU2 MBIST Control Register (STCU2_MB_CTRL15)	32	R/W	See section	86.6.45/ 4564

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
640	STCU2 MBIST Control Register (STCU2_MB_CTRL16)	32	R/W	See section	86.6.45/ 4564
644	STCU2 MBIST Control Register (STCU2_MB_CTRL17)	32	R/W	See section	86.6.45/ 4564
648	STCU2 MBIST Control Register (STCU2_MB_CTRL18)	32	R/W	See section	86.6.45/ 4564
64C	STCU2 MBIST Control Register (STCU2_MB_CTRL19)	32	R/W	See section	86.6.45/ 4564
650	STCU2 MBIST Control Register (STCU2_MB_CTRL20)	32	R/W	See section	86.6.45/ 4564
654	STCU2 MBIST Control Register (STCU2_MB_CTRL21)	32	R/W	See section	86.6.45/ 4564
658	STCU2 MBIST Control Register (STCU2_MB_CTRL22)	32	R/W	See section	86.6.45/ 4564
65C	STCU2 MBIST Control Register (STCU2_MB_CTRL23)	32	R/W	See section	86.6.45/ 4564
660	STCU2 MBIST Control Register (STCU2_MB_CTRL24)	32	R/W	See section	86.6.45/ 4564
664	STCU2 MBIST Control Register (STCU2_MB_CTRL25)	32	R/W	See section	86.6.45/ 4564
668	STCU2 MBIST Control Register (STCU2_MB_CTRL26)	32	R/W	See section	86.6.45/ 4564
66C	STCU2 MBIST Control Register (STCU2_MB_CTRL27)	32	R/W	See section	86.6.45/ 4564
670	STCU2 MBIST Control Register (STCU2_MB_CTRL28)	32	R/W	See section	86.6.45/ 4564
674	STCU2 MBIST Control Register (STCU2_MB_CTRL29)	32	R/W	See section	86.6.45/ 4564
678	STCU2 MBIST Control Register (STCU2_MB_CTRL30)	32	R/W	See section	86.6.45/ 4564
67C	STCU2 MBIST Control Register (STCU2_MB_CTRL31)	32	R/W	See section	86.6.45/ 4564
680	STCU2 MBIST Control Register (STCU2_MB_CTRL32)	32	R/W	See section	86.6.45/ 4564
684	STCU2 MBIST Control Register (STCU2_MB_CTRL33)	32	R/W	See section	86.6.45/ 4564
688	STCU2 MBIST Control Register (STCU2_MB_CTRL34)	32	R/W	See section	86.6.45/ 4564
68C	STCU2 MBIST Control Register (STCU2_MB_CTRL35)	32	R/W	See section	86.6.45/ 4564
690	STCU2 MBIST Control Register (STCU2_MB_CTRL36)	32	R/W	See section	86.6.45/ 4564
694	STCU2 MBIST Control Register (STCU2_MB_CTRL37)	32	R/W	See section	86.6.45/ 4564

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
698	STCU2 MBIST Control Register (STCU2_MB_CTRL38)	32	R/W	See section	86.6.45/ 4564
69C	STCU2 MBIST Control Register (STCU2_MB_CTRL39)	32	R/W	See section	86.6.45/ 4564
6A0	STCU2 MBIST Control Register (STCU2_MB_CTRL40)	32	R/W	See section	86.6.45/ 4564
6A4	STCU2 MBIST Control Register (STCU2_MB_CTRL41)	32	R/W	See section	86.6.45/ 4564
6A8	STCU2 MBIST Control Register (STCU2_MB_CTRL42)	32	R/W	See section	86.6.45/ 4564
6AC	STCU2 MBIST Control Register (STCU2_MB_CTRL43)	32	R/W	See section	86.6.45/ 4564
6B0	STCU2 MBIST Control Register (STCU2_MB_CTRL44)	32	R/W	See section	86.6.45/ 4564
6B4	STCU2 MBIST Control Register (STCU2_MB_CTRL45)	32	R/W	See section	86.6.45/ 4564
6B8	STCU2 MBIST Control Register (STCU2_MB_CTRL46)	32	R/W	See section	86.6.45/ 4564
6BC	STCU2 MBIST Control Register (STCU2_MB_CTRL47)	32	R/W	See section	86.6.45/ 4564
6C0	STCU2 MBIST Control Register (STCU2_MB_CTRL48)	32	R/W	See section	86.6.45/ 4564
6C4	STCU2 MBIST Control Register (STCU2_MB_CTRL49)	32	R/W	See section	86.6.45/ 4564
6C8	STCU2 MBIST Control Register (STCU2_MB_CTRL50)	32	R/W	See section	86.6.45/ 4564
6CC	STCU2 MBIST Control Register (STCU2_MB_CTRL51)	32	R/W	See section	86.6.45/ 4564
6D0	STCU2 MBIST Control Register (STCU2_MB_CTRL52)	32	R/W	See section	86.6.45/ 4564
6D4	STCU2 MBIST Control Register (STCU2_MB_CTRL53)	32	R/W	See section	86.6.45/ 4564
6D8	STCU2 MBIST Control Register (STCU2_MB_CTRL54)	32	R/W	See section	86.6.45/ 4564
6DC	STCU2 MBIST Control Register (STCU2_MB_CTRL55)	32	R/W	See section	86.6.45/ 4564
6E0	STCU2 MBIST Control Register (STCU2_MB_CTRL56)	32	R/W	See section	86.6.45/ 4564
6E4	STCU2 MBIST Control Register (STCU2_MB_CTRL57)	32	R/W	See section	86.6.45/ 4564
6E8	STCU2 MBIST Control Register (STCU2_MB_CTRL58)	32	R/W	See section	86.6.45/ 4564
6EC	STCU2 MBIST Control Register (STCU2_MB_CTRL59)	32	R/W	See section	86.6.45/ 4564

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6F0	STCU2 MBIST Control Register (STCU2_MB_CTRL60)	32	R/W	See section	86.6.45/ 4564
6F4	STCU2 MBIST Control Register (STCU2_MB_CTRL61)	32	R/W	See section	86.6.45/ 4564
6F8	STCU2 MBIST Control Register (STCU2_MB_CTRL62)	32	R/W	See section	86.6.45/ 4564
6FC	STCU2 MBIST Control Register (STCU2_MB_CTRL63)	32	R/W	See section	86.6.45/ 4564
700	STCU2 MBIST Control Register (STCU2_MB_CTRL64)	32	R/W	See section	86.6.45/ 4564
704	STCU2 MBIST Control Register (STCU2_MB_CTRL65)	32	R/W	See section	86.6.45/ 4564
708	STCU2 MBIST Control Register (STCU2_MB_CTRL66)	32	R/W	See section	86.6.45/ 4564
70C	STCU2 MBIST Control Register (STCU2_MB_CTRL67)	32	R/W	See section	86.6.45/ 4564
710	STCU2 MBIST Control Register (STCU2_MB_CTRL68)	32	R/W	See section	86.6.45/ 4564
714	STCU2 MBIST Control Register (STCU2_MB_CTRL69)	32	R/W	See section	86.6.45/ 4564
718	STCU2 MBIST Control Register (STCU2_MB_CTRL70)	32	R/W	See section	86.6.45/ 4564
71C	STCU2 MBIST Control Register (STCU2_MB_CTRL71)	32	R/W	See section	86.6.45/ 4564
720	STCU2 MBIST Control Register (STCU2_MB_CTRL72)	32	R/W	See section	86.6.45/ 4564
724	STCU2 MBIST Control Register (STCU2_MB_CTRL73)	32	R/W	See section	86.6.45/ 4564
728	STCU2 MBIST Control Register (STCU2_MB_CTRL74)	32	R/W	See section	86.6.45/ 4564
72C	STCU2 MBIST Control Register (STCU2_MB_CTRL75)	32	R/W	See section	86.6.45/ 4564
730	STCU2 MBIST Control Register (STCU2_MB_CTRL76)	32	R/W	See section	86.6.45/ 4564
734	STCU2 MBIST Control Register (STCU2_MB_CTRL77)	32	R/W	See section	86.6.45/ 4564

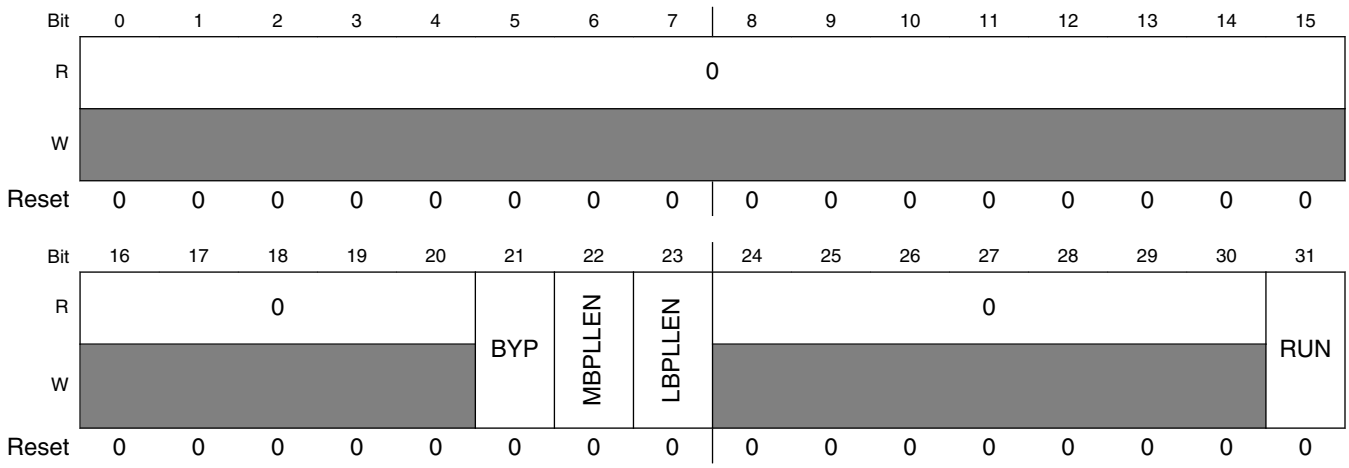
86.6.1 STCU2 Run Register (STCU2_RUN)

The STCU2_RUN register defines the RUN bit to start the off-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Register description

Address: 0h base + 0h offset = 0h



STCU2_RUN field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BYP	Bypass mode 0 Off-Line Self-Test is active. In case the RUN bit into the STCU2_RUN register is not set before the Hard-Coded WDG Time-out, the WDTO error is generated into the STCU2_ERR_STAT register. 1 Off-Line Self-Test is completely bypassed and the access to STCU2 is locked until the STCU2_SCK register is written according to the request access mode.
22 MBPLLEN	Off-Line MBIST with PLL Enabled 0 Off-Line MBIST is executed without using the on-chip PLL. 1 Off-Line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
23 LBPLLEN	Off-Line LBIST with PLL Enabled 0 Off-Line LBIST is executed without using the on-chip PLL. 1 Off-Line LBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUN	RUN: RUN The RUN bit is automatically cleared by the STCU2 when the off-line self testing procedure has been completed. 0 Idle 1 Off-Line Self testing procedure is running

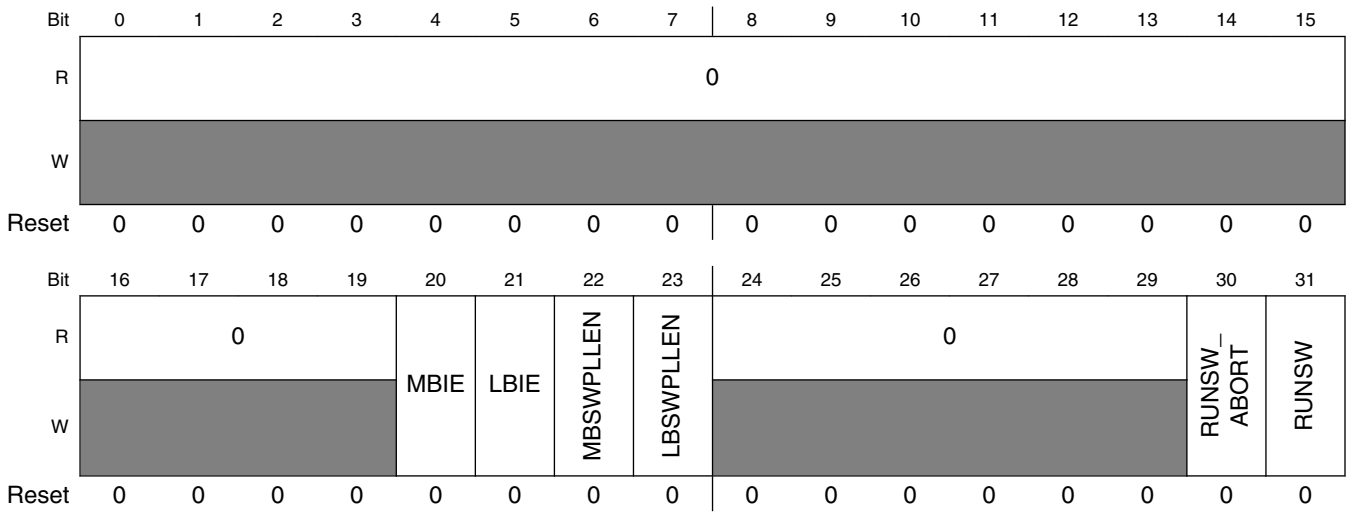
86.6.2 STCU2 Run Software Register (STCU2_RUNSW)

The STCU2_RUNSW register defines the RUN bit to start the on-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU2_CFG[WRP] = 0
- On-line Self-Test phase is active. The On-Line condition starts when the Off-Line is over. In this condition, system is alive and the SW can program the STCU2

Address: 0h base + 4h offset = 4h



STCU2_RUNSW field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 MBIE	MBIST Interrupt Enable 0 Interrupt is not generated at the end of the software MBIST execution phase of the selected Memories. 1 At the end of the software MBIST execution phase of the concurrent selected memories the STCU2 forces an interrupt request to notify that MBISTs of concurrent selected memories are over. In case there is a sequence of concurrent runs in the same Self-Test run, the interrupt request is generated at the end of each concurrent runs provided that the software will clear the STCU2_INT_FLG[MBIFG] bit in between.
21 LBIE	LBIST Interrupt Enable 0 Interrupt is not generated at the end of the software LBIST execution phase of the selected LBIST. 1 At the end of the software LBIST execution phase of the concurrent selected LBIST the STCU2 forces an interrupt request to notify that LBISTs of the concurrent selected partitions are over. In case there is a sequence of concurrent runs in the same Self-Test run, the interrupt request is generated at the end of each concurrent run provided that the software will clear the STCU2_INT_FLG.LBIFG bit in between.
22 MBSWPLLEN	On-Line MBIST with PLL Enabled 0 On-Line MBIST is executed without using the on-chip PLL. 1 On-Line MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.

Table continues on the next page...

STCU2_RUNSW field descriptions (continued)

Field	Description
23 LBSWPLLEN	On-Line LBIST with PLL Enabled 0 On-Line LBIST is executed without using the on-chip PLL. 1 On-Line LBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RUNSW_ABORT	Software abort of the self-test currently running This bit is automatically cleared by the STCU2 when the software self-test abort sequence has been completed. 0 On-line self-test abort is not requested. 1 On-line self-test abort is requested.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the on-line self testing procedure has been completed. 0 Idle 1 On-line self testing procedure is running

86.6.3 STCU2 SK Code Register (STCU2_SKC)

The STCU2_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus) or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU2_SKC register
- write the key2 into the STCU2_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

It should be noted that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU2_CFG), the SSCM interface is no longer available.

In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

In case the STCU2 register access lasts more cycles than the one defined into the Hard-coded WDG time-out, the STCU2 write access is locked and the WDG and Register ITF clocks are switched off. Also in this case, in order to enable again the write access to the STCU2 and the WDG and Register ITF clocks, it is required to apply again the sequence.

In case it is required to extend the STCU2 register access cycles before the Hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, Key1 has not to be applied otherwise a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

The STCU2_SKC register is not readable. The value 00000000h is always returned in case of read operation.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SKC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_SKC field descriptions

Field	Description
0–31 SKC	STCU2 security key code for off-line Test = D3FEA98Bh: Key1 to unlock the write access the STCU2 = 2C015674h: Key2 to unlock the write access the STCU2 STCU2 security key code for on-line Test = 753F924Eh: Key1 to unlock the write access the STCU2 = 8AC06DB1h: Key2 to unlock the write access the STCU2

86.6.4 STCU2 Configuration Register (STCU2_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

The access to this register is described in the following figure. It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the on-line Self-Test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

MBIST can be run in three different test types according to how the STCU2_CFG register bits 27 and 28 are set.

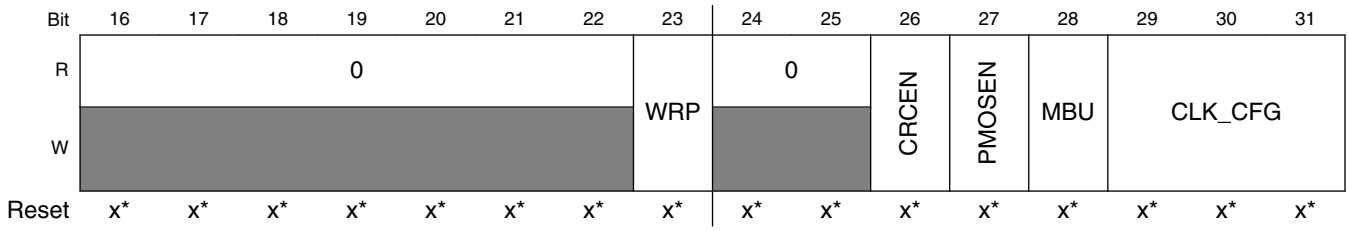
The following table describes the test algorithm and coverage of the different test types.

Table 86-1. MBIST algorithm and coverage according to test type

MBIST Type	STCU2_CFG[PMOSEN]	STCU2_CFG[MBU]	MBIST Algorithm	Internal Nodes/ Circuits Additionally Covered by Test	Usecase	Test Cycle
Full Test	1	0	Test memory using all built in algorithms. Open PMOS algorithm included	Address Decoder and Bitcell as well as resistive defects in the address decoder	Used by FSL production test. Recommended to use in the field for MBIST online self-test.	About 3 to 4x more than Auto Test
Reduced Test	0	0	Test memory using all built in algorithms except the open PMOS algorithm.	Address Decoder and Bitcell	Recommended to use in the field for MBIST online self-test if the Full Test takes too long.	About 10% less than Full Test
Auto Test	x	1	A smaller set of algorithms which target latent defect mechanisms.	Latent defects such as NBTI of the PMOS transistors in the bitcells	Recommended to use for MBIST offline self-test as an optimum balance of test time vs. fault coverage.	About 3 to 4x faster than Full Test

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PTR							LB_DELAY							
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_CFG field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 PTR	LBIST or MBIST pointer PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0 then the LBIST0 is initially scheduled. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1): pointer to MBIST 7Fh: pointer to NIL. No BIST execution.
8–15 LB_DELAY	Delay LBIST run LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. The allowed delay time are the following: 00h: No delay 01h: 1 x16 STCU2 Core clock cycles 02h: 2 x 16 STCU2 Core clock cycles 03h: 3 x 16 STCU2 Core clock cycles ... FDh: 253 x 16 STCU2 Core clock cycles FEh: 254 x 16 STCU2 Core clock cycles FFh: 255 x 16 STCU2 Core clock cycles Note that in case the Frequency of the STCU2 core clock is 16 MHz, the delay parameters is expressed in µs. **NOTE:** This field should be written with 0xFF for self test run. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
23 WRP	Write Protection 0: Specific STCU2 registers can be written through IPS bus interface after Off-Line Self-Test sequence has been completed 1: Specific STCU2 registers cannot be written though IPS also after Off-Line Self-Test sequence has been completed preventing in such a way any user application write operation
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 CRCEN	CRCEN: CRC Enable comparison You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 0 The CRC comparison is not performed and the STCU2_ERR_STAT.CRCS[SW] and STCU2_ERR_STAT.UF/RF global flags are not updated in case of mismatches between STCU2.CRCE and STCU2.CRCR values 1 The CRC comparison is performed and the status is updated into the related flags of the STCU2_ERR_STAT register
27 PMOSEN	MBIST PMOS Test Enable When MBU is 1 then "Auto test" MBIST test type will be selected regardless of the setting of this field.. 0 When MBU is 0 then selects the "Reduced test" MBIST test type. 1 When MBU is 0 then selects the "Full test" MBIST test type.
28 MBU	MBIST MBU Test Enabled 0 Either "Full test" or "Reduced test" MBIST test type will be selected according to the PMOSEN value. 1 Selects "Auto test" MBIST test type regardless of the PMOSEN value.
29–31 CLK_CFG	Logic, Memory Bist and STCU2 core CLK Clock configuration CLK_CFG defines the ratio between the sys_clk and the TCK used to program both the LBIST and the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are the following: 000: sys_clk 001: sys_clk/2 010: sys_clk/3 ... 101: sys_clk/6 110: sys_clk/7 111: sys_clk/8 You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.

86.6.5 STCU2 PLL Configuration Register (STCU2_PLL_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_PLL_CFG register defines the parameters used to program the PLL only when the Off-line L/MBIST are performed and STCU2_RUN[MBPLLEN] = 1 or STCU2_RUN[LBPLLEN] = 1.

In the On-line condition, these bits are not effective. It is also possible to set the PLL during active On-line mode to prevent any potential functionality issue.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		PLLODF						0					PLLIDF			0									PLLLDF						
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_PLL_CFG field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 PLLODF	PLL Output Division Factor The value of this field drives the Output Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLODF value and the related PLL's Output Division factor.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 PLLIDF	PLL Input Division Factor The value of this field drives the Input Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLIDF value and the related PLL's Input Division factor.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PLLLDF	PLL Loop Division Factor The value of this field drives the Loop Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLLDF value and the related PLL's Loop Division factor.

86.6.6 STCU2 Watchdog Register Granularity (STCU2_WDG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_WDG register defines the time budget of LBIST and MBIST execution providing a protection mechanism in case of dead-lock or end-less conditions during the Self-Test procedure. When the Off-line Self-Test sequence is not run and the BYP bit of the STCU2_CFG register is not set, the bits 15...0 defines the time-out before the bit WDTO of the STCU2_ERR register is set and the STCU2 core clock is switched off.

In case Off/On-line Self Test sequence is run, it defines the time-out of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDGEOC																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

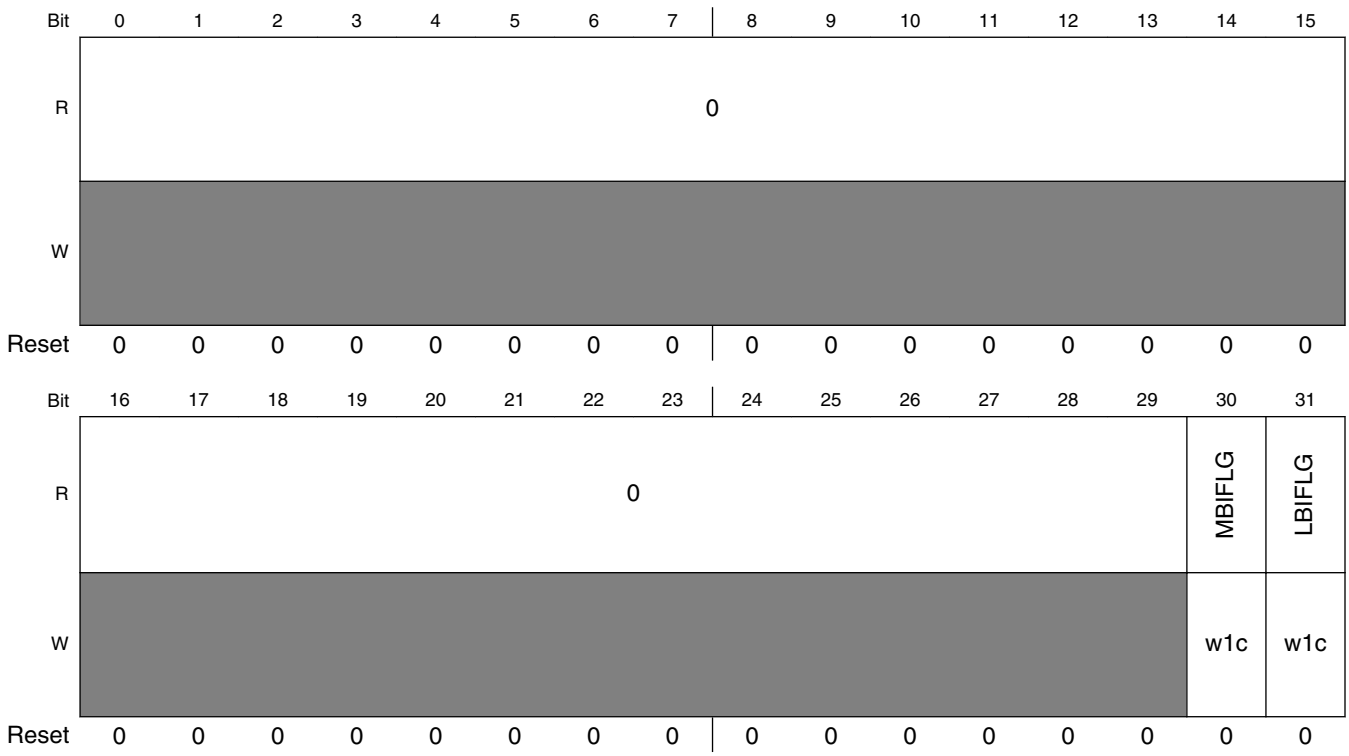
STCU2_WDG field descriptions

Field	Description
0–31 WDGEOC	Watchdog End of Count Timer This value has to be set in order to define the time budget related to the Off/On-Line Self Test execution and check that everything is correctly working within this slot of time. The allowed delay time are the following: 0000 0000h: 1 x 16 STCU2 Core clock cycles 0000 0001h: 2 x 16 STCU2 Core clock cycles 0000 0002h: 3 x 16 STCU2 Core clock cycles ... FFFF FFFDh: 4294967294 x 16 STCU2 Core clock cycles FFFF FFFEh: 4294967295 x 16 STCU2 Core clock cycles FFFF FFFFh: 4294967296 x 16 STCU2 Core clock cycles

86.6.7 STCU2 Interrupt Flag Register (STCU2_INT_FLG)

The STCU2_INT_FLG register includes an MBIST interrupt pending bit and an LBIST interrupt pending bit. Each bit is effective only during the On-Line Self-Test phase and is managed only when the related control bit in the STCU2_RUNSW register (MBIE or LBIE) is enabled.

Address: 0h base + 18h offset = 18h



STCU2_INT_FLG field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MBIFLG	MBIST Interrupt Flag 0 No interrupt is pending 1 An interrupt, highlighting that the On-Line execution of the concurrent scheduled MBIST is completed, is pending. Write 1 to clear the bit.
31 LBIFLG	LBIST Interrupt Flag 0 No interrupt is pending 1 An interrupt, highlighting that the On-Line execution of the concurrent scheduled LBIST is completed, is pending. Write 1 to clear the bit.

86.6.8 STCU2 CRC Expected Status Register (STCU2_CRCE)

NOTE

See the chip specific information for the default reset value of this field.

The STCU2_CRCE register includes the expected signature of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU2_CRCE_RES and computes the CRC signature of the STCU2's critical signals and when STCU2_CFG[CRCEN] = 1, it is able to provide the Self-Check capability of the STCU2 managing the CRCS[SW] bits of the STCU2_ERR_STAT register. The polynomial CRC used to evaluate the status of this register is the following:

$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$

CRC-32 [ethernet protocol]

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRCE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

STCU2_CRCE field descriptions

Field	Description
0–31 CRCE	CRC expected signature

86.6.9 STCU2 CRC Read Status Register (STCU2_CRCR)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CRCR register reports the value obtained at the end of the Off/On-line Self-Test sequence. It might be used for diagnoses and also as additional check with respect to the CRCS[SW] bit of the STCU2_ERR_STAT register.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCR																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

STCU2_CRCR field descriptions

Field	Description
0–31 CRCR	Read CRC signature

86.6.10 STCU2 Error Register (STCU2_ERR_STAT)

The STCU2_ERR_STAT register includes the status flags related to the STCU2 internal fault conditions occurred during the configuration or the On/Off-Line self testing execution.

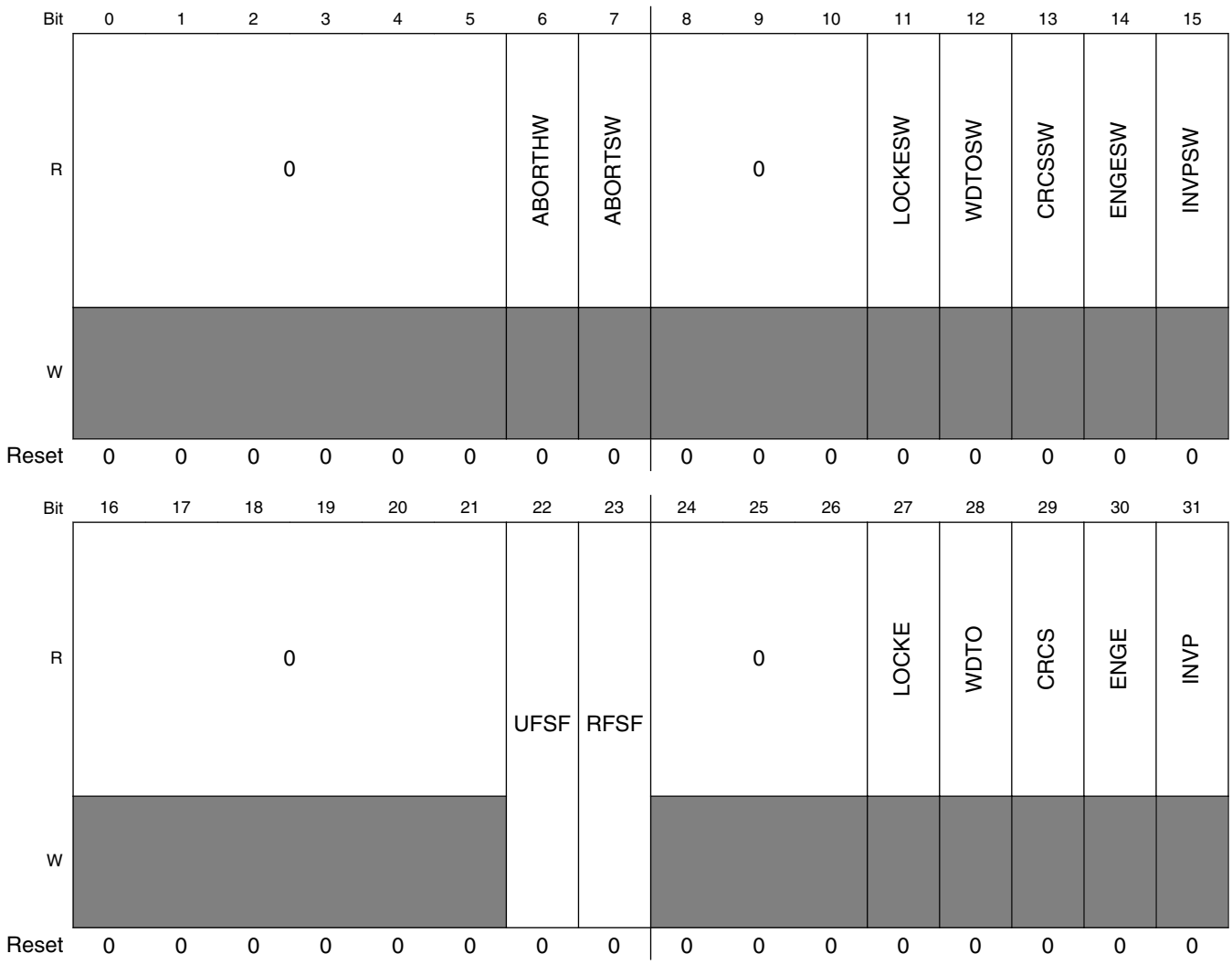
The UFSF and RFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the following figure and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

Register description

Address: 0h base + 24h offset = 24h



STCU2_ERR_STAT field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ABORTHW	On-line hardware abort flag You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 No hardware abort was requested during the On-Life Self-Test sequence 1 A hardware abort was detected during the On-Life Self-Test sequence
7 ABORTSW	On-line software abort flag You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 No software abort was requested during the On-Life Self-Test sequence 1 A software abort was detected during the On-Life Self-Test sequence

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 LOCKESW	On-Line LOCK Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 In case PLL is enabled, it is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the On-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUNSW.LBSWPLLEN is set and/or during the MBIST run when STCU2_RUNSW[MBSWPLLEN] = 1.
12 WDTOSW	On-Line Watchdog time-out You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
13 CRCSSW	On-Line CRC Status This flag is activated ONLY when On-Line Self Test is performed and STCU2_CFG.CRCEN is set to 1 otherwise it is always forced to 0 preventing in such a way the generation of the related STCU2_ERR_STAT.UFSF/RFSF bits. In both the cases, the content of the STCU2_CRCR register report the current evaluated CRC. You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Successful CRC comparison or comparison has been masked (STCU2_CFG.CRCEN set to 0) 1 Failed CRC comparison
14 ENGESW	On-Line Engine Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	On-Line Invalid Pointer You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit:

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	• Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. 0 No errors that trigger the Unrecoverable Faults condition 1 There are errors that trigger the Unrecoverable Faults condition
23 RFSF	Recoverable Faults Status Flag 0 No errors that trigger the Recoverable Faults condition 1 There are errors that trigger the Recoverable Faults condition
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKE	Off-Line LOCK Error 0 The PLL is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlights that there has been an unexpected PLL unlock event during the Off-Line Self-Test sequence execution. The Off-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUN[LBPLLEN] = 1 and/or during the MBIST run when STCU2_RUN[MBPLLEN] = 1.
28 WDTO	Off-Line Watchdog time-out 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • The STCU2_RUN[RUN] bit or the STCU2_CFG[BYP] bit are not set before the Watchdog reach the End of Count • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
29 CRCS	Off-Line CRC status This flag is activated ONLY when Off-Line Self Test is performed STCU2_CFG[CRCEN] is set to 1 otherwise it is always forced to 0 preventing in such a way the generation of the related STCU2_ERR_STAT.UFSF/RFSF bits. In both the cases, the content of the STCU2_CRCR register report the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU2_CFG[CRCEN] programmed to 0) 1 Failed CRC comparison
30 ENGE	Off-Line Engine Error 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	• Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Off-Line Invalid pointer 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)

86.6.11 STCU2 Error FM Register (STCU2_ERR_FM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_ERR_FM register defines the fault mapping of the STCU2 faults described into the register STCU2_ERR_STAT in terms of Unrecoverable or Recoverable Fault. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_ERR_FM field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
28 WDTOUFM	WatchDog time-out Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
29 CRCSUFM	CRC Status Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Mapping

86.6.12 STCU2 Off-Line LBIST Status Register (STCU2_LBS)

The STCU2_LBS register includes the results corresponding to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_LBS field descriptions

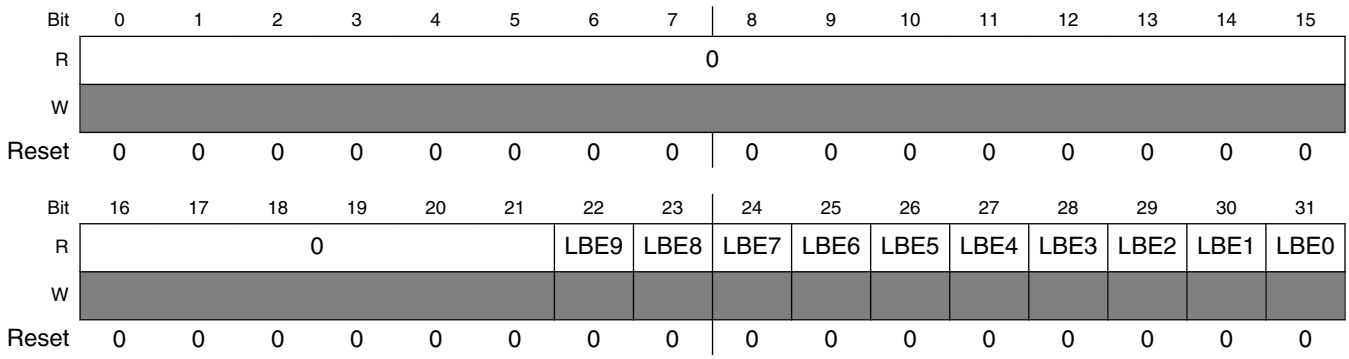
Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LBS9	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
23 LBS8	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
24 LBS7	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
25 LBS6	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
26 LBS5	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
27 LBS4	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
28 LBS3	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
29 LBS2	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBS1	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBS0	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

86.6.13 STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)

The STCU2_LBE register includes the End Flag related to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 30h offset = 30h



STCU2_LBE field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LBE9	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
23 LBE8	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
24 LBE7	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
25 LBE6	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
26 LBE5	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
27 LBE4	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
28 LBE3	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBE2	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

Table continues on the next page...

STCU2_LBE field descriptions (continued)

Field	Description
30 LBE1	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBE0	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

86.6.14 STCU2 On-Line LBIST Status Register (STCU2_LBSSW)

The STCU2_LBSSW register includes the results corresponding to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						LBSSW9	LBSSW8	LBSSW7	LBSSW6	LBSSW5	LBSSW4	LBSSW3	LBSSW2	LBSSW1	LBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_LBSSW field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LBSSW9	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
23 LBSSW8	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
24 LBSSW7	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
25 LBSSW6	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
26 LBSSW5	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
27 LBSSW4	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
28 LBSSW3	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
29 LBSSW2	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBSSW1	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBSSW0	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

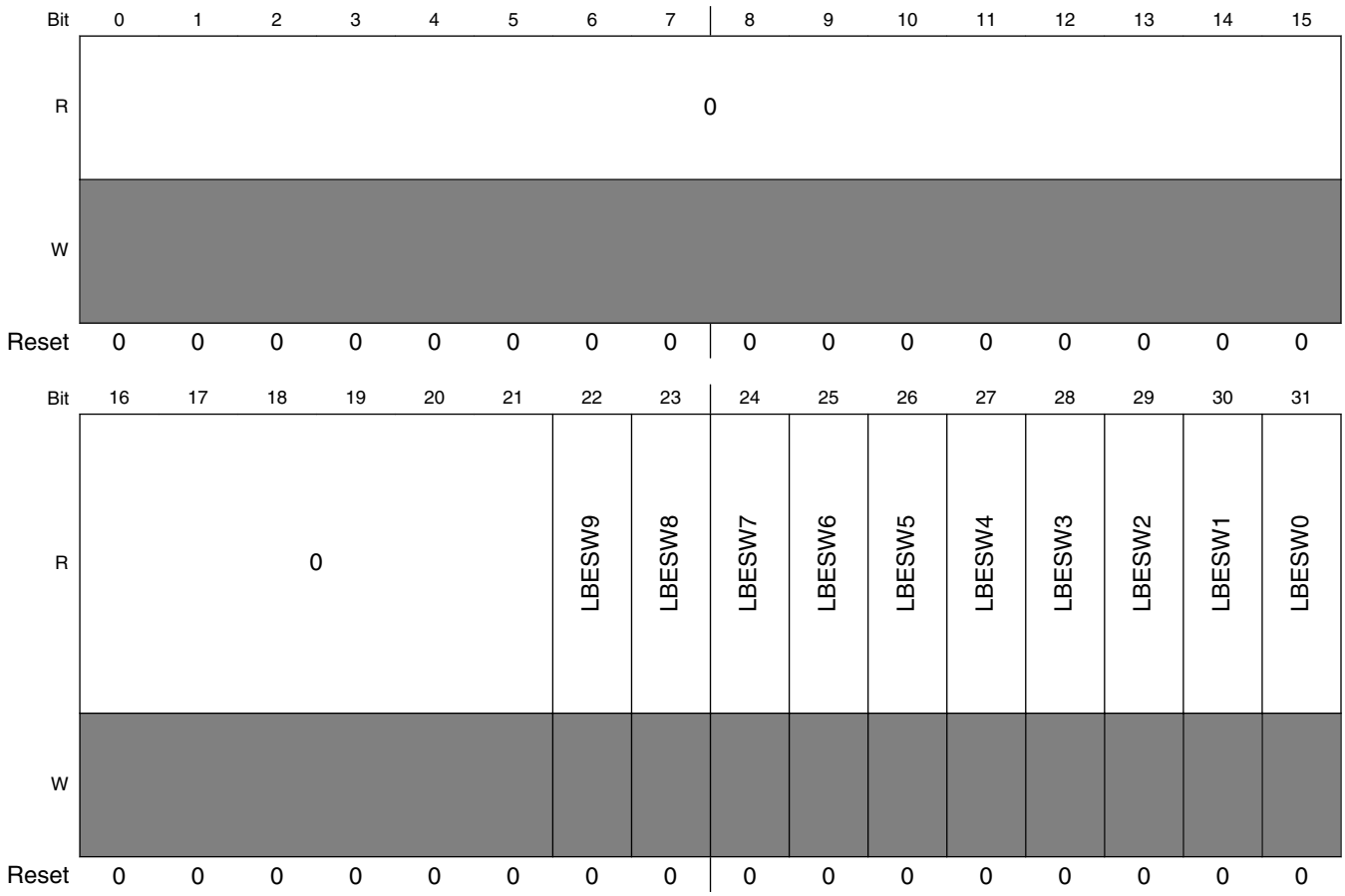
86.6.15 STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)

The STCU2_LBESW register includes the End Flag related to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 38h offset = 38h



STCU2_LBESW field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LBESW9	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
23 LBESW8	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
24 LBESW7	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished

Table continues on the next page...

STCU2_LBESW field descriptions (continued)

Field	Description
25 LBESW6	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
26 LBESW5	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
27 LBESW4	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
28 LBESW3	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBESW2	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBESW1	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBESW0	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished

86.6.16 STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

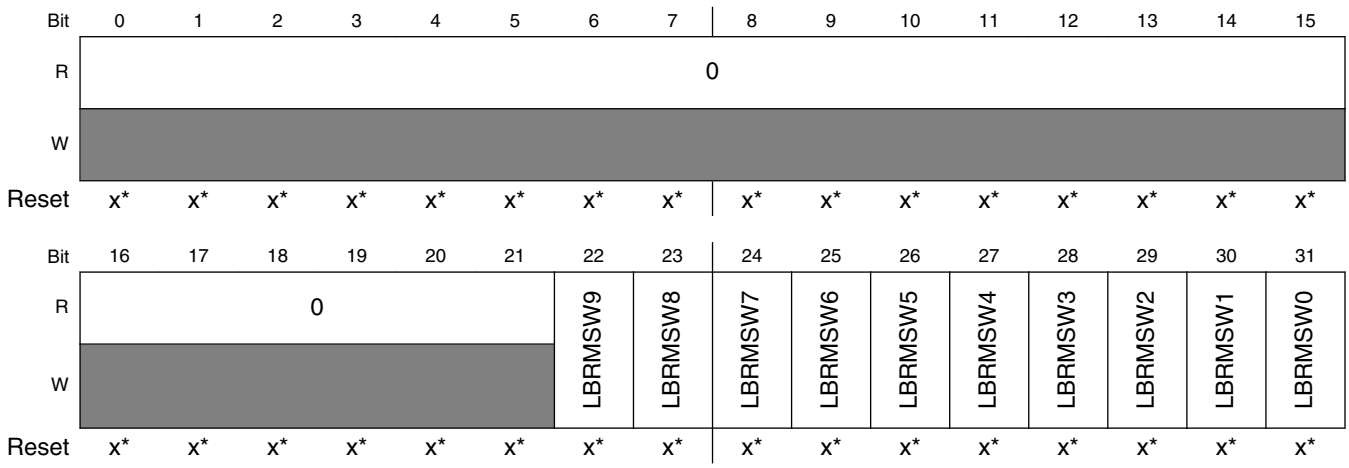
The STCU2_LBRMSW register defines the On-Line reset mapping for each LBIST. Global reset can be programmed.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 3Ch offset = 3Ch



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBRMSW field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LBRMSW9	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
23 LBRMSW8	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
24 LBRMSW7	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
25 LBRMSW6	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run

Table continues on the next page...

STCU2_LBRMSW field descriptions (continued)

Field	Description
26 LBRMSW5	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
27 LBRMSW4	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
28 LBRMSW3	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
29 LBRMSW2	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
30 LBRMSW1	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
31 LBRMSW0	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run

86.6.17 STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LBUFM register defines the fault mapping of each LBIST in terms of Unrecoverable or Recoverable Fault.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W	Reserved															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved						LBUFM9	LBUFM8	LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
W	Reserved															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBUFM field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22 LBUFM9	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 LBUFM8	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 LBUFM7	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 LBUFM6	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 LBUFM5	LBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_LBUFM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 LBUFM4	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 LBUFM3	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 LBUFM2	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 LBUFM1	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 LBUFM0	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

86.6.18 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)

The STCU2_MBSL register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31	MBS30	MBS29	MBS28	MBS27	MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSL field descriptions

Field	Description
0 MBS31	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS30	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS29	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBS28	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS27	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS26	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS25	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS24	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS23	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS22	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS21	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS20	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS19	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
13 MBS18	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS17	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS16	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS15	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS14	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS13	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS12	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS11	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS10	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS9	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS8	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS7	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS6	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS5	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS4	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS3	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS2	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS1	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS0	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

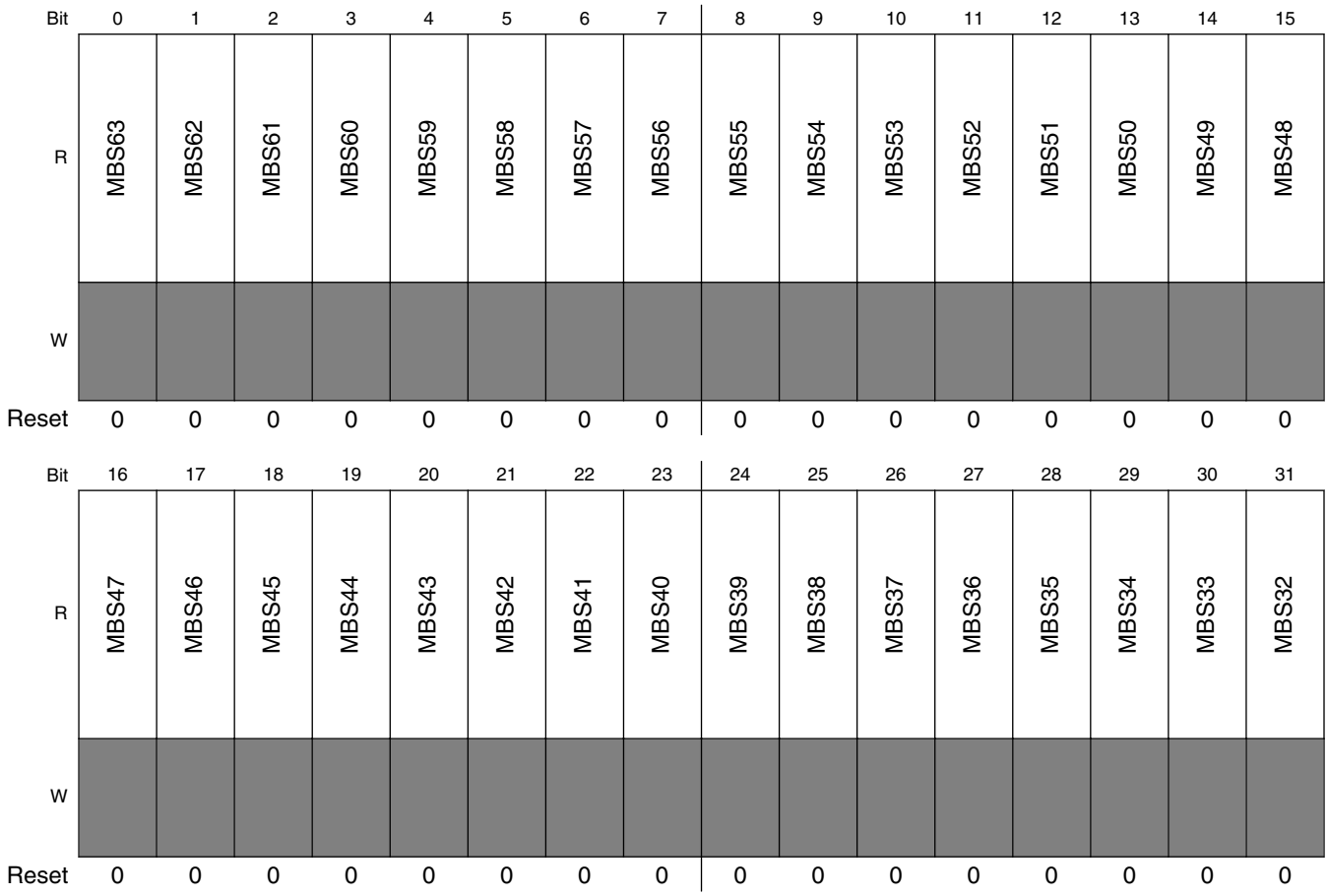
86.6.19 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)

The STCU2_MBSM register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs .

Register description

Address: 0h base + 48h offset = 48h



STCU2_MBSM field descriptions

Field	Description
0 MBS63	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS62	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS61	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
3 MBS60	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS59	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS58	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS57	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS56	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS55	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS54	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS53	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS52	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS51	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS50	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS49	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS48	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS47	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS46	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS45	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS44	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS43	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS42	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS41	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS40	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS39	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS38	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

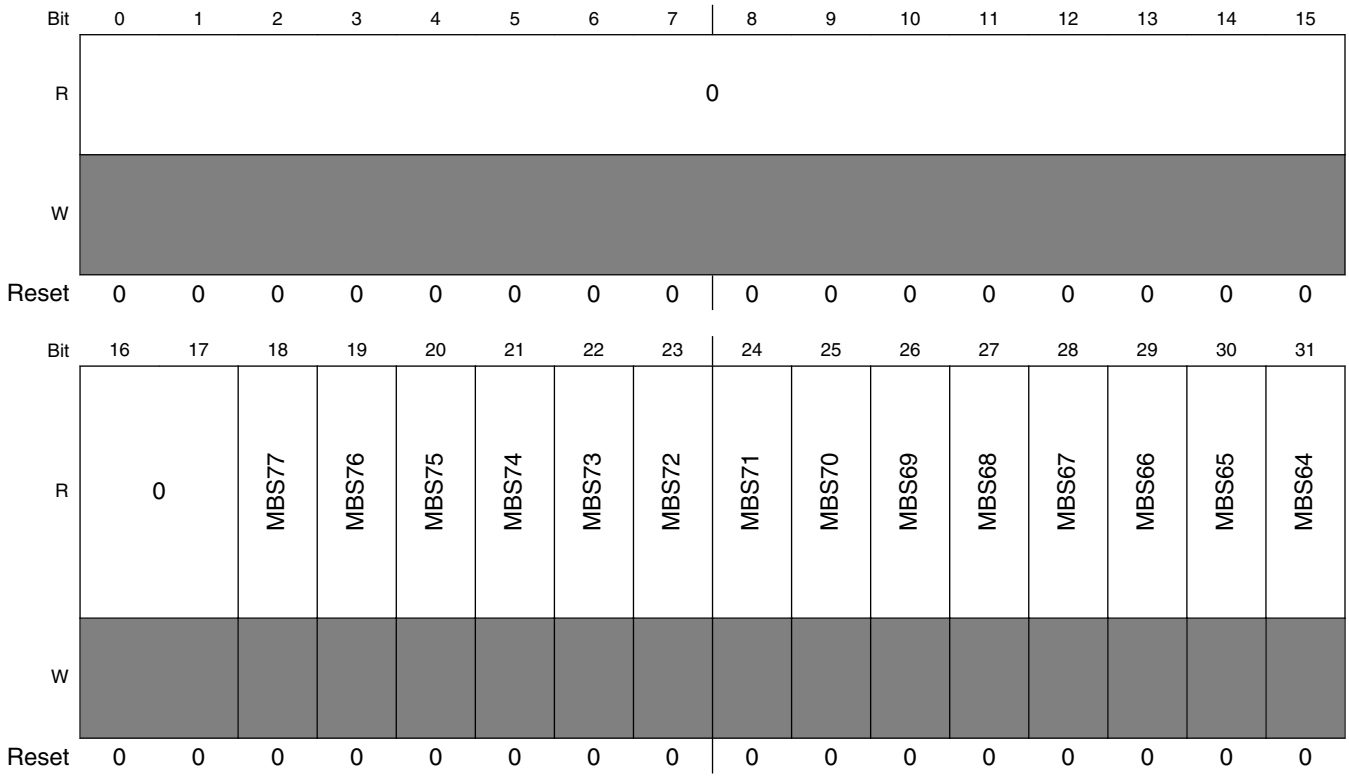
Field	Description
26 MBS37	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS36	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS35	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS34	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS33	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS32	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

86.6.20 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)

The STCU2_MBSH register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 4Ch offset = 4Ch



STCU2_MBSH field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MBS77	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS76	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS75	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS74	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBSH field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS73	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS72	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS71	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS70	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS69	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS68	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS67	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSH field descriptions (continued)

Field	Description
29 MBS66	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS65	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS64	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

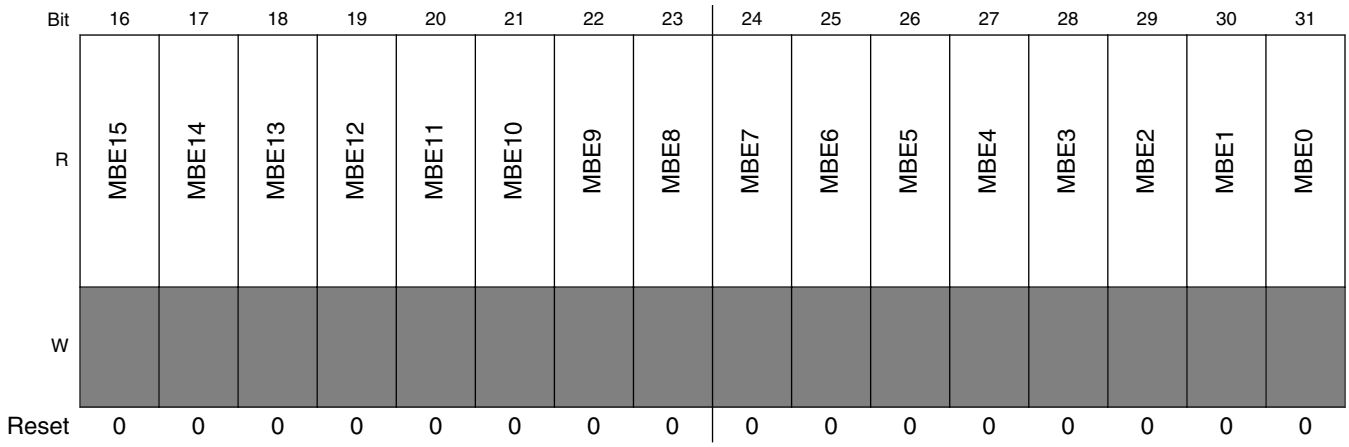
86.6.21 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)

The STCU2_MBEL register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEL field descriptions

Field	Description
0 MBE31	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE30	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBE29	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE28	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE27	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE26	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE25	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE24	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE23	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE22	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE21	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE20	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE19	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE18	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE17	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE16	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE15	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE14	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE13	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE12	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE11	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE10	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE9	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE8	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE7	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE6	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE5	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE4	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE3	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE2	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE1	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE0	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

86.6.22 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)

The STCU2_MBEM register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE63	MBE62	MBE61	MBE60	MBE59	MBE58	MBE57	MBE56	MBE55	MBE54	MBE53	MBE52	MBE51	MBE50	MBE49	MBE48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE47	MBE46	MBE45	MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEM field descriptions

Field	Description
0 MBE63	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE62	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
2 MBE61	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE60	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE59	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE58	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE57	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE56	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE55	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE54	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE53	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE52	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE51	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE50	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
14 MBE49	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE48	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE47	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE46	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE45	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE44	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE43	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE42	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE41	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE40	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE39	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE38	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

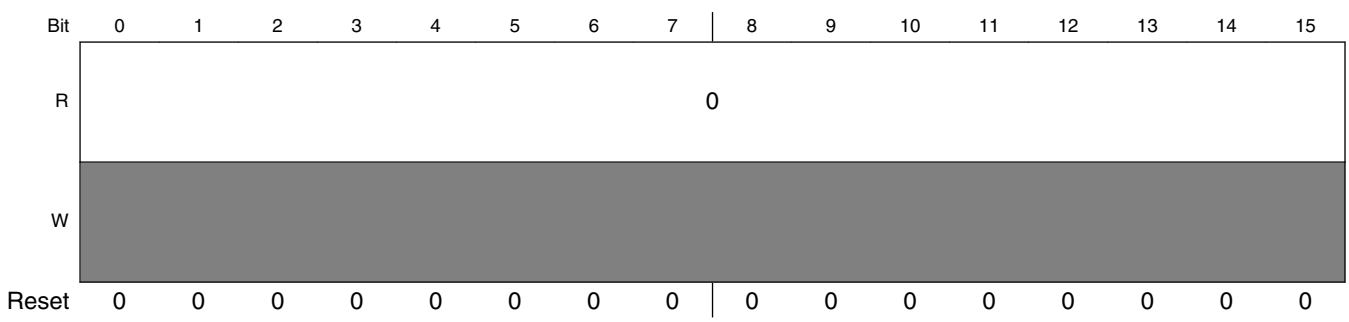
Field	Description
26 MBE37	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE36	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE35	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE34	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE33	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE32	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

86.6.23 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)

The STCU2_MBEH register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 58h offset = 58h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		MBE77	MBE76	MBE75	MBE74	MBE73	MBE72	MBE71	MBE70	MBE69	MBE68	MBE67	MBE66	MBE65	MBE64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEH field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MBE77	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE76	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE75	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE74	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE73	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE72	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE71	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE70	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEH field descriptions (continued)

Field	Description
26 MBE69	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE68	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE67	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE66	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE65	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE64	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

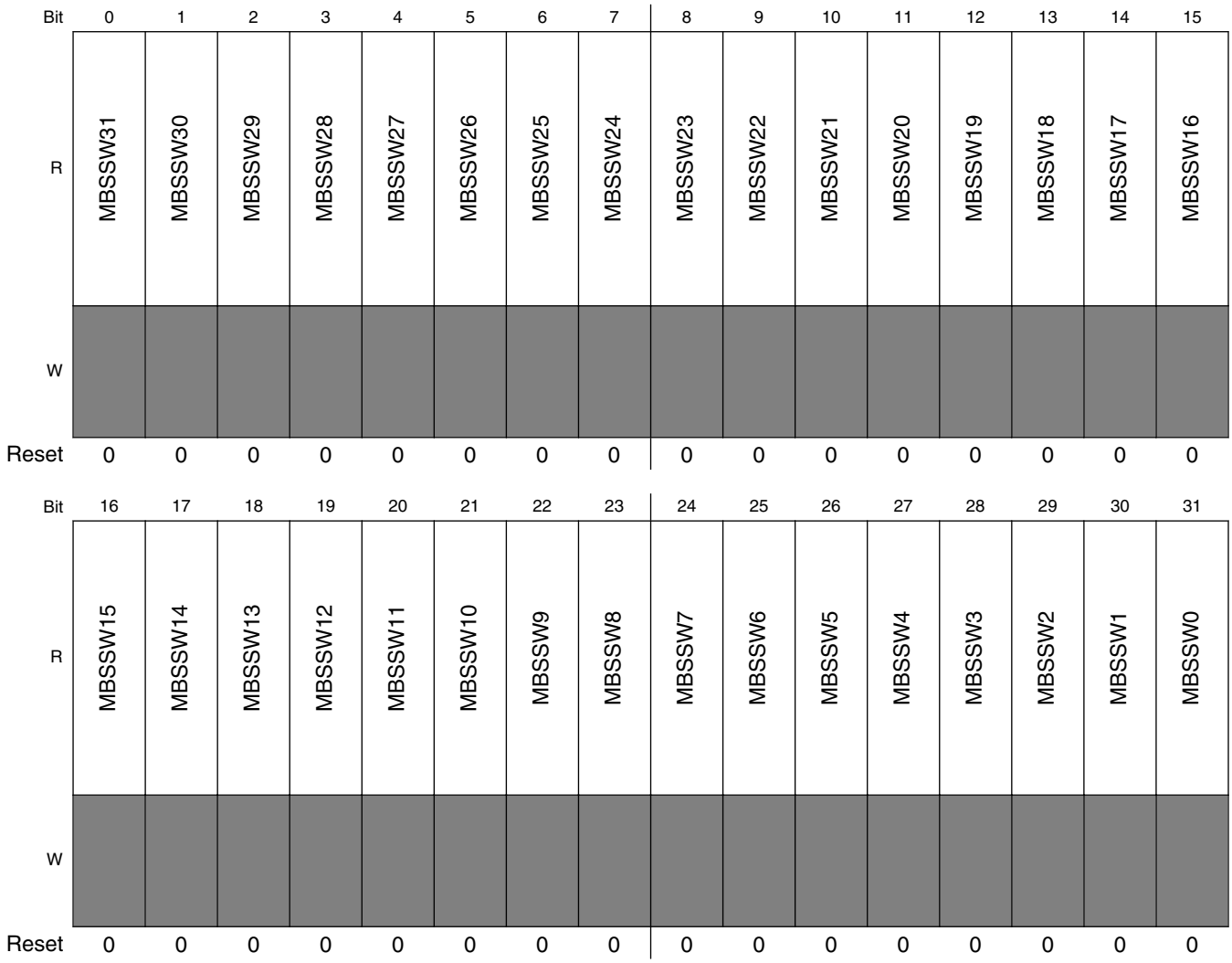
86.6.24 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)

The STCU2_MBSLSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 5Ch offset = 5Ch



STCU2_MBSLSW field descriptions

Field	Description
0 MBSSW31	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW30	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW29	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW28	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW27	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW26	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW25	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW24	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW23	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW22	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW21	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW20	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW19	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW18	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW17	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW16	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW15	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW14	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW13	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW12	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW11	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW10	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW9	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW8	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW7	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW6	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW5	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW4	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW3	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW2	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW1	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW0	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

86.6.25 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)

The STCU2_MBSMSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63	MBSSW62	MBSSW61	MBSSW60	MBSSW59	MBSSW58	MBSSW57	MBSSW56	MBSSW55	MBSSW54	MBSSW53	MBSSW52	MBSSW51	MBSSW50	MBSSW49	MBSSW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSMSW field descriptions

Field	Description
0 MBSSW63	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW62	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW61	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW60	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW59	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW58	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW57	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW56	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
8 MBSSW55	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW54	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW53	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW52	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW51	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW50	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW49	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW48	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW47	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW46	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW45	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW44	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
20 MBSSW43	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW42	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW41	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW40	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW39	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW38	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW37	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW36	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW35	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW34	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW33	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW32	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

86.6.26 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)

The STCU2_MBSHSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		MBSSW77	MBSSW76	MBSSW75	MBSSW74	MBSSW73	MBSSW72	MBSSW71	MBSSW70	MBSSW69	MBSSW68	MBSSW67	MBSSW66	MBSSW65	MBSSW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSHSW field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MBSSW77	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSHSW field descriptions (continued)

Field	Description
19 MBSSW76	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW75	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW74	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW73	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW72	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW71	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW70	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW69	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW68	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW67	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW66	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW65	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSHSW field descriptions (continued)

Field	Description
31 MBSSW64	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

86.6.27 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)

The STCU2_MBELSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW31	MBESW30	MBESW29	MBESW28	MBESW27	MBESW26	MBESW25	MBESW24	MBESW23	MBESW22	MBESW21	MBESW20	MBESW19	MBESW18	MBESW17	MBESW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBELSW field descriptions

Field	Description
0 MBESW31	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW30	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW29	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW28	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW27	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW26	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW25	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW24	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
8 MBESW23	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW22	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW21	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW20	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW19	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW18	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW17	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW16	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW15	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW14	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW13	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW12	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
20 MBESW11	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW10	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW9	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW8	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW7	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW6	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW5	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW4	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW3	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW2	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW1	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW0	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

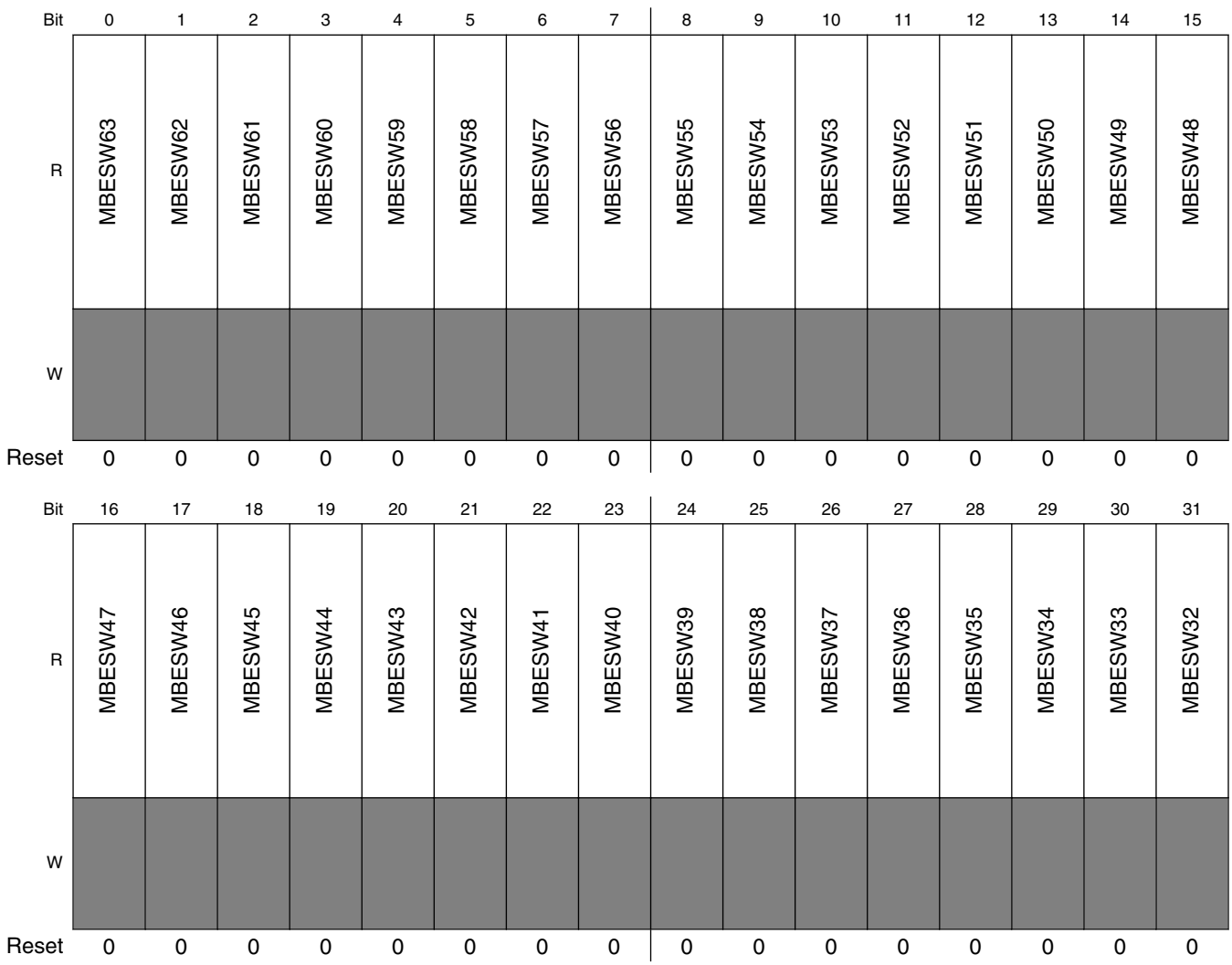
86.6.28 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)

The STCU2_MBEMSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 6Ch offset = 6Ch



STCU2_MBEMSW field descriptions

Field	Description
0 MBESW63	On-Line End status (32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW62	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW61	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW60	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW59	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW58	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW57	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW56	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBESW55	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW54	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW53	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW52	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW51	On-Line End status (32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW50	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW49	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW48	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW47	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW46	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW45	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW44	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW43	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW42	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW41	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW40	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW39	On-Line End status (32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW38	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW37	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW36	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW35	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW34	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW33	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW32	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

86.6.29 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)

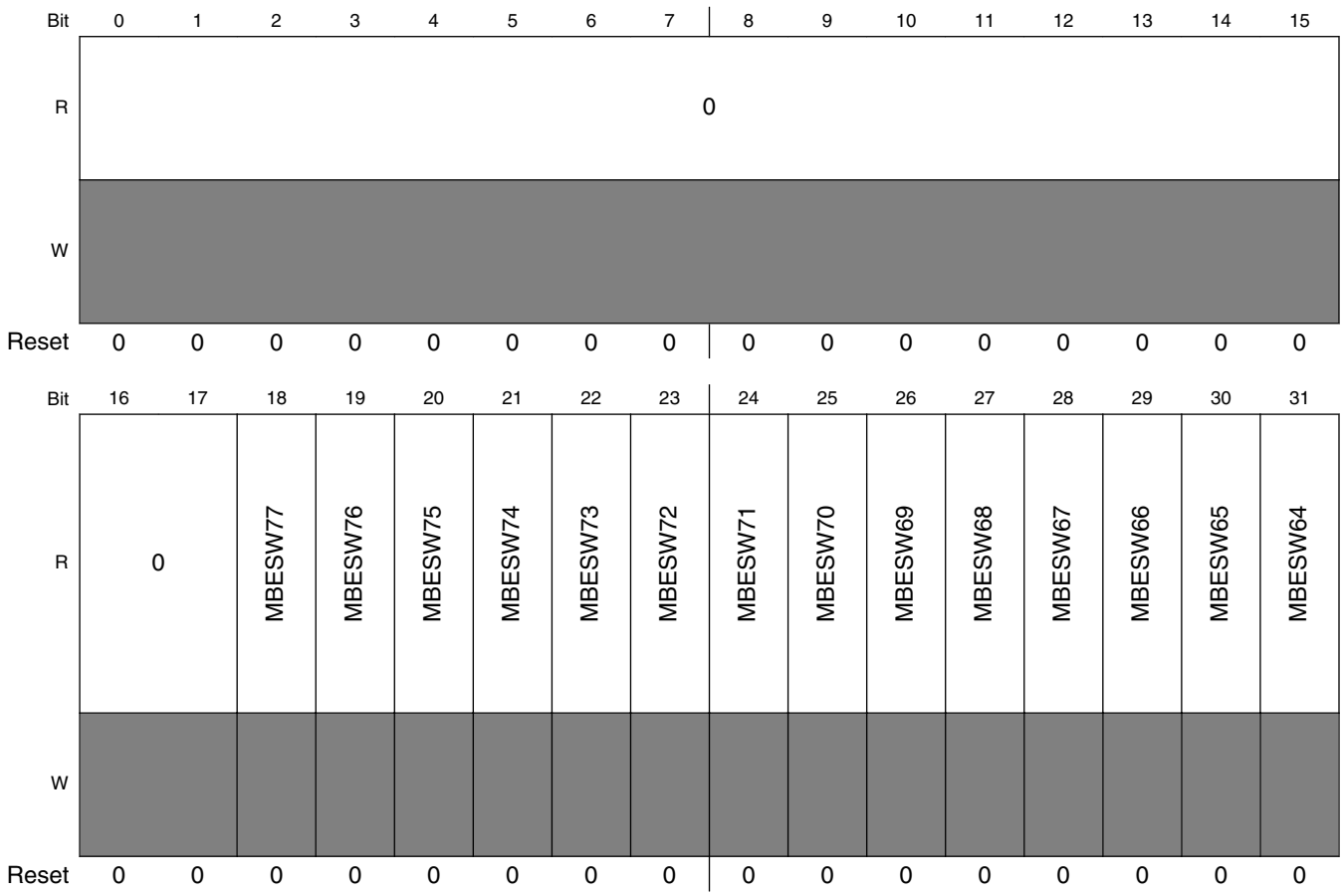
The STCU2_MBEHSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Register description

Address: 0h base + 70h offset = 70h



STCU2_MBEHSW field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MBESW77	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW76	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW75	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW74	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW73	MBESWx: On-Line End status (64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBEHSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW72	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW71	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW70	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW69	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW68	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW67	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW66	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW65	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW64	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

86.6.30 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFML register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFML field descriptions

Field	Description
0 MBUFM31	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM30	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM29	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM28	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
4 MBUFM27	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM26	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM25	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM24	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM23	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM22	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM21	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM20	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM19	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM18	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM17	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM16	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
16 MBUFM15	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM14	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM13	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM12	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM11	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM10	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM9	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM8	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM7	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM6	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM5	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM4	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
28 MBUFM3	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM2	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM1	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM0	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

86.6.31 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMM register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MBUFM63	MBUFM62	MBUFM61	MBUFM60	MBUFM59	MBUFM58	MBUFM57	MBUFM56	MBUFM55	MBUFM54	MBUFM53	MBUFM52	MBUFM51	MBUFM50	MBUFM49	MBUFM48
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MBUFM47	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMM field descriptions

Field	Description
0 MBUFM63	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM62	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM61	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM60	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM59	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM58	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM57	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM56	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM55	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
9 MBUFM54	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM53	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM52	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM51	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM50	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM49	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM48	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM47	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM46	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM45	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM44	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM43	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
21 MBUFM42	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM41	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM40	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM39	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM38	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM37	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM36	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM35	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM34	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM33	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM32	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

86.6.32 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMH register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		MBUFM77	MBUFM76	MBUFM75	MBUFM74	MBUFM73	MBUFM72	MBUFM71	MBUFM70	MBUFM69	MBUFM68	MBUFM67	MBUFM66	MBUFM65	MBUFM64
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMH field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MBUFM77	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMH field descriptions (continued)

Field	Description
19 MBUFM76	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM75	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM74	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM73	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM72	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM71	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM70	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM69	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM68	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM67	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM66	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM65	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMH field descriptions (continued)

Field	Description
31 MBUFM64	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

86.6.33 STCU2 LBIST Control Register (STCU2_LB_CTRL*n*)

NOTE

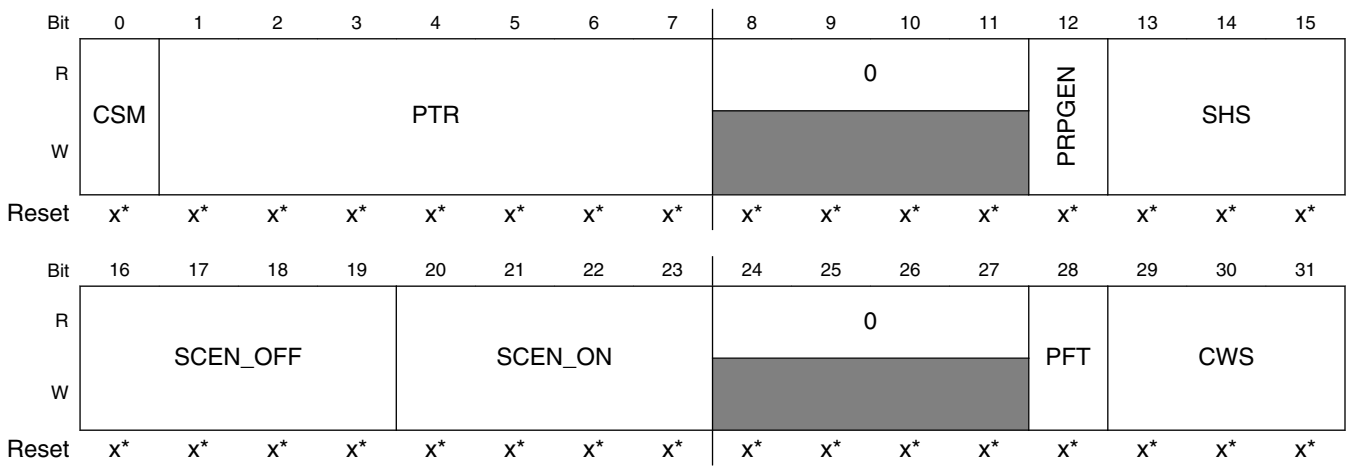
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_CTRL register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 100h offset + (64d × i), where i=0d to 9d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode The next LBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise, it is scheduled sequentially to the completion of the current LBIST execution.

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
	0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST pointer PTR defines the logical pointer to the next LBIST. In case of NIL pointer the CSM bit has to be set Sequential (0) to define this is the end of the list. The self testing procedure is stopped when the current LBIST has been completed. LBIST POINTERS 0000b - pointer to LBIST partition 0 0001b - pointer to LBIST partition 1 0010b - pointer to LBIST partition 2 0011b - pointer to LBIST partition 3 0100b - pointer to LBIST partition 4 0101b - pointer to LBIST partition 5 0110b - pointer to LBIST partition 6 0111b - pointer to LBIST partition 7 1000b - pointer to LBIST partition 8 1001b - pointer to LBIST partition 9 ... 0x7F - pointer to NIL, end of LBIST execution Values not defined in this list will result in an error being set in the STCU2_ERR register.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 PRPGEN	Enable PRPG Loading 0 Default LBIST value of the PRPG is used during LBIST run 1 LBIST PRPG value is initialized with the STCU2_LB_PRPGL/H
13–15 SHS	Shift speed SHS defines the shift speed 000 Shift at full rate of STCU2 core clock 001 Shift at 1/2 rate of STCU2 core clock 010 Shift at 1/3 rate of STCU2 core clock 011 Shift at 1/4 rate of STCU2 core clock 100 Shift at 1/5 rate of STCU2 core clock 101 Shift at 1/6 rate of STCU2 core clock 110 Shift at 1/7 rate of STCU2 core clock 111 Shift at 1/8 rate of STCU2 core clock
16–19 SCEN_OFF	Scan enable OFF SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN **NOTE:** Scen_off must be programmed to a value >=1 0000 0 delay cycles 0001 1 delay cycle

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
	0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
20–23 SCEN_ON	Scan enable ON SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN **NOTE:** Scen_on delay register value must be programmed to a value >=1> 0000 0 delay cycles 0001 1 delay cycle 0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 PFT	Past Flush Test The LBIST controller, by default, applies 32 Flush Test patterns. This bit allows to skip/apply the Flush test pattern sequence. 0 Apply Flush Test Patterns 1 Skip Flush Test Patterns
29–31 CWS	Capture window size CWS defines the capture window size. 000 illegal

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
	001 controller waits 1 shift cycle for capture to finish 010 controller waits 2 shift cycles for capture to finish 011 controller waits 3 shift cycles for capture to finish 100 controller waits 4 shift cycles for capture to finish 101 controller waits 5 shift cycles for capture to finish 110 controller waits 6 shift cycles for capture to finish 111 controller waits 7 shift cycles for capture to finish

86.6.34 STCU2 LBIST PC Stop Register (STCU2_LB_PCS*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 104h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PCS																									
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PCS*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

86.6.35 STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PRPGL register defines the LSB part (31-0) of the PRPG start value of the LBIST controller.

The size of the register depends on the number of PRPG bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 108h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRPGx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PRPGL*n* field descriptions

Field	Description
0–31 PRPGx	PRPG LBIST Low Bits This field defines the Low part (31-0) of the LBIST PRPG start value.

86.6.36 STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PRPGH register defines the MSB part (63-32) of the PRPG start value of the LBIST controller.

The size of the register depends on the number of PRPG bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 10Ch offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRPGx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PRPGH*n* field descriptions

Field	Description
0–31 PRPGx	PRPG LBIST High Bits This field is the High part (63-32) of the LBIST PRPG start value.

86.6.37 STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREL register defines the LSB part (31-0) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 110h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISREx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREL*n* field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected low Bits

STCU2_LB_MISREL*n* field descriptions (continued)

Field	Description
	This field defines the low part (31..0) of the Expected MISR.

86.6.38 STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREH register defines the MSB part (63-32) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 114h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISREx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREH*n* field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

86.6.39 STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRL register reports the LSB part (31-0) of the MISR obtained at the end of the Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 118h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRL*n* field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read Low Bits This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution

86.6.40 STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRH register report the MSB part (63-32) of the MISR obtained at the end of each Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 11Ch offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRH*n* field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read High Bits

STCU2_LB_MISRRH*n* field descriptions (continued)

Field	Description
	This field is equivalent to the High Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution

86.6.41 STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRELSW register defines the LSB part (31-0) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 120h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISRESWx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRELSW*n* field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

86.6.42 STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREHSW register defines the MSB part (63-32) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 124h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISRESWx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREHSW*n* field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

86.6.43 STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRLSW register report the LSB part (31-0) of the MISR obtained at the end of the On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 128h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSWx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRLSW*n* field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read Low Bin This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution.

86.6.44 STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRHSW register report the MSB part (63-32) of the MISR obtained at the end of each On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 12Ch offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSWx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRHSW*n* field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read High Bits This field is equivalent to the Low Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution.

86.6.45 STCU2 MBIST Control Register (STCU2_MB_CTRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

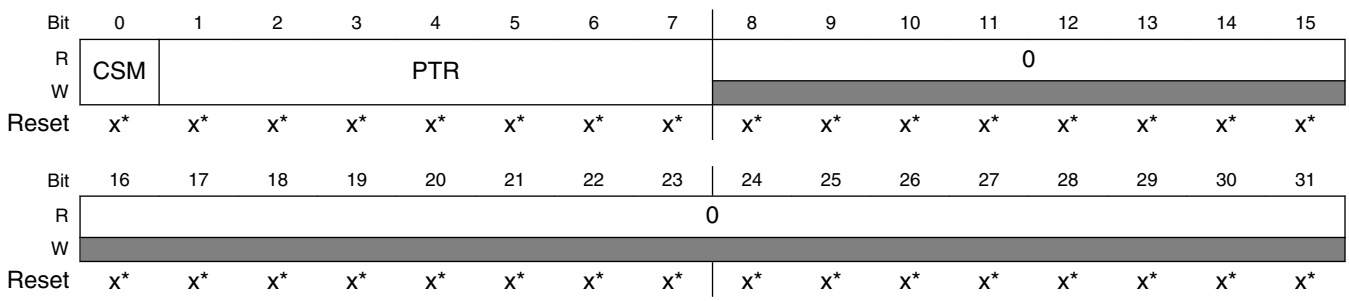
The STCU2_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

The offset of this register is a function of NMCUT .

Address: 0h base + 600h offset + (4d × i), where i=0d to 77d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST orMBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit must be set Sequential (0) to define this is the end of the list. The self-testing procedure is stopped when the current MBIST has been completed. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1b): pointer to MBIST 7Fh: pointer to NIL. No BIST execution. others: invalid pointer => an error is set into the STCU2_ERR register.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

86.7 Use cases and limitations

The STCU2 module is designed to give high flexibility during the on/off-line test. In the following are reported the programming operations that have to be performed to correctly use the functionality of this module considering the three typical usage cases:

- the Off-line Self Test Sequence after a reset trigger event
- the bypass mode of the Self Test Sequence after a reset trigger event
- the On-line Self-test Sequence

86.7.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and L/MBIST execution parameters stored into the NVM memory.The target is to cover the amount of physical defects in the digital logic and in the System RAMs/ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU2_LBUFM register
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU2_MBUFML/M/H registers
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU2_ERR_FM register
- Program the STCU2_LB_CTRL, STCU2_LB_PCS, STCU2_MISREL/H registers of each LBIST to be executed. Optionally, also the STCU2_LB_PRPGL/H can be programmed in order to load a specific seed.
- Program the STCU2_MB_CTRL registers of each NMCUT to be executed
- Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution
- Program the STCU2_CFG register to define: the core and LBIST/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, the CRCREN bit to enable the self-checking feature and finally set the pointer to the first LBIST/NMCUT to be executed

- In case the internal CRC comparison has been enabled, program the CRCE value expected at the end of the off-line Self Test sequence into the STCU2_CRCE register
- In case the PLL usage has been enabled, program the STCU2_PLL_CFG value in order to set the required system frequency
- Program the RUN bit into the STCU2_RUN register to enable the Off-Line Self-Test execution and the L/MBPLLEN bits to enable the PLL during L/MBIST execution according to STCU2_PLL_CFG register content
- Start the Off-Line Self-Test and, depending on PLL management and L/MBIST execution order, when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register.
- The STCU2 switchs off the core clock at the end of the Self-Test run and releases the PLL control in case this feature has been enabled.
- The STCU2 resets the STCU2_RUN[RUN] bit and the device exits from boot sequence. In case there are failures, the STCU2 flags these failures toward the FCCU.
- The software application reads the STCU2_LBS flag register to check the failed LBIST when UF or RF conditions is/are detected.
- The software application reads the STCU2_LBE flag register to check the LBIST still on-going when UF or RF is/are detected.
- The software application reads the STCU2_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
- The software application reads the STCU2_MBEL/M/H flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.
- The software application reads the bits INV, ENGE, CRCS, WDTO, LOCKE of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.
- In case the CRCEN bit has been enabled, the software application read the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU2_ERR register.
- The software application reads for each device's LBIST, the STCU2_LBMISREL/H and STCU2_LBIST_MISRRL/H registers to check the coherence with the STCU2_LBS bits.

NOTE

The software application reads operations on the STCU2 CRC and LBIST MISR registers after off-line self test execution has been successfully performed are just an additional reliability layer added to improve the already implemented Auto Self-Test feature included into STCU2.

86.7.2 Online self-test sequence

This is the typical mode of using the STCU2 module during the application run. The IPS interface is used to program the STCU2 registers and to schedule the L/MBIST execution. Since the On-Line L/MBIST operations clean-up the content of the related Digital Logic (LBIST) or RAM (MBIST) the activation of this mode has to be done carefully. The suggested sequence is the following:

- Unlock the On-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Check and/or clean-up the bit WRP into the STCU2_CFG register to open the IPS access on the STCU2 On-Line Self-Test registers
- The LBIST, MBIST and STCU2 internal fault reaction should have been already programmed during the Off-Line Self-Test sequence and so they should not be programmed again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten setting respectively the registers: STCU2_LBUFM, STCU2_MBUFML/M/H and STCU2_ERR_FM.
- Program the Reset management during on-Line LBIST execution setting the STCU2_LBRMSW register
- The STCU2_LB_CTRL, STCU2_LB_PCS should have been already programmed during Off-Line Self-test sequence and so they should not be programmed again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten. The registers STCU2_LB_MISRELSW/HSW have to be programmed according to the expected signature.
- Overwrite/Program the STCU2_MB_CTRL registers of each NMCUT to be executed
- Overwrite/Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution

- Overwrite/Program the STCU2_CFG register in order to define: the core and L/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, the CRCREN bit to enable the self-checking feature and finally set the pointer to the first LBIST/NMCUT to be executed.
- In case the internal CRC comparison has been enabled, overwrite/program the CRCE value expected at the end of the on-line Self Test sequence into the STCU2_CRCE register
- Program the RUNSW bit into the STCU2_RUNSW register to enable the On-Line Self-Test execution; the L/MBSWPLLEN bits to enable the PLL lock signal monitor during L/MBIST execution and the L/MBIE to enable the related Interrupt requests.
- Start the On-Line Self-Test and depending on PLL management and L/MBIST execution order, also when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register
- In case the L/MBIE have been enabled, an interrupt is generated at the end of all the L/MBIST execution. The software application has to manage these interrupt signals and clean-up the related flag (L/MBIFLG) into the STCU2_INT_FLG register.
- The STCU2 switch-off the core clock at the end of the Self-Test run
- The functional reset activated is managed by the STCU2_LBRMSW.
- In case, at least one of the selected LBIST enables the Global functional reset, the STCU2 requests a functional reset which restarts the system while, when all the selected LBIST enables its own dedicated functional reset, only these local reset lines are applied and only the related LBIST partitions are cleaned-up.
- The STCU2 resets the STCU2_RUNSW[RUNSW] bit. In both the reset cases and in case of fault/s, the STCU2 flags the failure toward the FCCU.
- The software application reads the STCU2_LBSSW flag register to check the failed LBIST when UF or RF conditions is/are detected
- The software application reads the STCU2_LBESW flag register to check if the LBIST is still on-going when UF or RF is/are detected
- The software application reads the STCU2_MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected

- The software application reads the STCU2_MBELSW/MSW/HSW flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected
- The software application reads the bit INVPSW, ENGESW, CRCSSW, WDTOSW, LOCKESW of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected
- In case the CRCEN bit has been enabled, the software application reads the CRCE and CRCR registers to check the coherence with the bit CRCSSW of the STCU2_ERR register
- The software application reads for each device's LBIST, the STCU2_LBMISRELSW/HSW and STCU2_LBIST_MISRRLSW/HSW registers to check the coherence with the STCU2_LBS bits

Note

The software application read operations on the STCU2 CRC and LBIST MISR registers after on-line self-test execution has been successfully performed are just an additional reliability layer to be added to improve the already implemented Auto Self-Test feature included in the STCU2.

Note

Flash memory must be in the idle state when BIST execution is started. Program and erase functionality must not be enabled when entering BIST execution.

86.7.3 Bypass USER Mode

In this case, the USER application parameters stored in FLASH and read by SSCM are written in order to skip the Self-Test procedure after a reset trigger event is applied. It might be useful in case the device is not configured for applications requiring improved reliability. The sequence is the following:

1. Unlock the STCU2 access writing the Off-Line Key1/Key2 sequence into the STCU2_SKC register
2. Set the BYP bit in the STCU2_RUN register

After BYP bit is set, the core clock is switched off, the Self-Test sequence is not applied and the system can wake up and start the user application.

86.7.4 Design implementation information

The following list reports the limitations related to the current implementation:

- Every NMCUT BIST can be run once during the online or offline self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- Because software has full control of the system while applications are running during the online self-test, STCU2 is not allowed to take control of the PLL but can just monitor the lock signal to flag wrong unlock events.

CAUTION

Do not execute an LBIST on more than one partition at a time.

Chapter 87
Register Protection (REG_PROT)

87.1 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE). This is shown in the following figure.

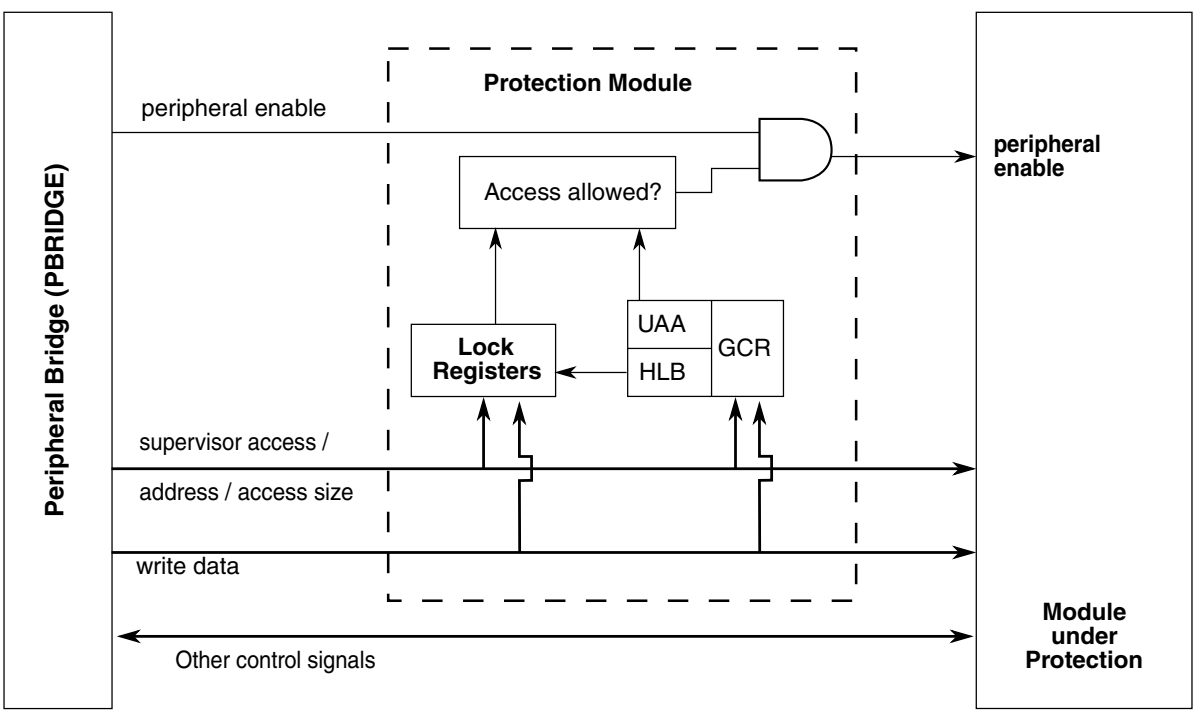


Figure 87-1. REG_PROT Block Diagram

87.2 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

87.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

87.4 External signal description

There are no external signals.

87.5 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in the following figure.

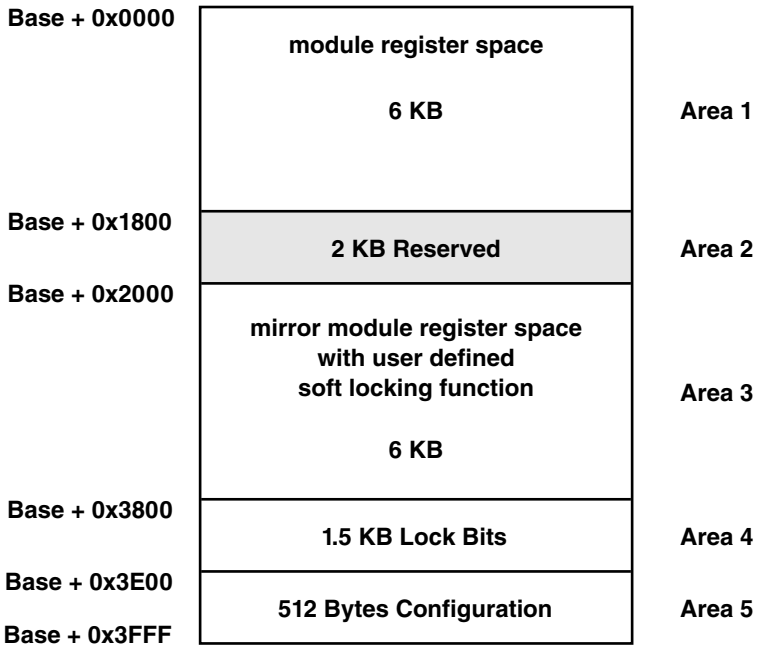


Figure 87-2. REG_PROT Memory Diagram

- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an automatic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.
- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

87.5.1 Memory map

Following is the memory map for a module which is protected via a REG_PROT module.

Table 87-1. Generic Protected Module Memory Map

Offset	Use
0x0000	Module Register 0 (MR0)
0x0001	Module Register 1 (MR1)
0x0002	Module Register 2 (MR2)
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)
0x1800 - 0x1FFF	Reserved
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 -Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143
0x3E00 - 0x3FFB	Reserved
0x3FFC	Global Configuration Register (GCR)

Note

Reserved registers in area #1 will be handled as specified in the protected module's documentation.

87.5.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

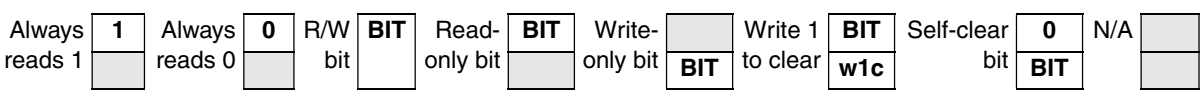


Figure 87-3. Register description

87.5.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

87.5.2.2 Module register and set soft lock bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBRn.SLBm, according to the mapping described in the Soft Lock Bit Register section.

87.6 Memory map and registers

REG_PROT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3800	Soft Lock Bit Register n (REG_PROT_SLBRn)	8	R/W	00h	87.6.1/4575
3FFC	Global Configuration Register (REG_PROT_GCR)	32	R/W	0000_0000h	87.6.2/4577

87.6.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area #1.

Table 87-2. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6

Table continues on the next page...

Table 87-2. Soft Lock Bits vs. Protected Address (continued)

Soft Lock Bit	Protected address
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

NOTE

- As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.
- Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3800h offset = 3800h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0	0	SLB0	SLB1	SLB2	SLB3
Write	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

REG_PROT_SLBRn field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB) WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB) WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB) WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB) WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MRn register

Table continues on the next page...

REG_PROT_SLBRn field descriptions (continued)

Field	Description
	SLB0 can block accesses to MR[n *4 + 0] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MRn register SLB1 can block accesses to MR[n *4 + 1] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MRn register SLB2 can block accesses to MR[n *4 + 2] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MRn register SLB3 can block accesses to MR[n *4 + 3] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable

87.6.2 Global Configuration Register (REG_PROT_GCR)

NOTE

Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3FFCh offset = 3FFCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0							UAA	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REG_PROT_GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit This register cannot be cleared once it is set by software. It can only be cleared by a system reset.

Table continues on the next page...

REG_PROT_GCR field descriptions (continued)

Field	Description
	NOTE: Access in User Mode is read-only; in Supervisor Mode access is read/write. 1 All SLB bits are write protected and can not be modified. 0 All SLB bits are accessible and can be modified.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

87.7 Functional description

The following sections describe the functional characteristics of the Register Protection module.

87.7.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depends on the SoC and/or module.

For all addresses that are protected there are SLBRn.SLBm bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBRn.SLBm bit is always 0b0 no matter what software is writing to.

87.7.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBRn.SLBm bit needs to be changed. This can be done using the following methods:

- Modify the SLBRn.SLBm directly by writing to area #4
- Set the SLBRn.SLBm bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

87.7.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBRn.SLBm bit has a mask bit SLBRn.WEm which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

The following figure shows two modification examples. In the left example there is a write access to the SLBRn register specifying a mask value which allows modification of all SLBRn.SLBm bits. The example on the right specifies a mask which only allows modification of the bits SLBRn.SLB[3:1].

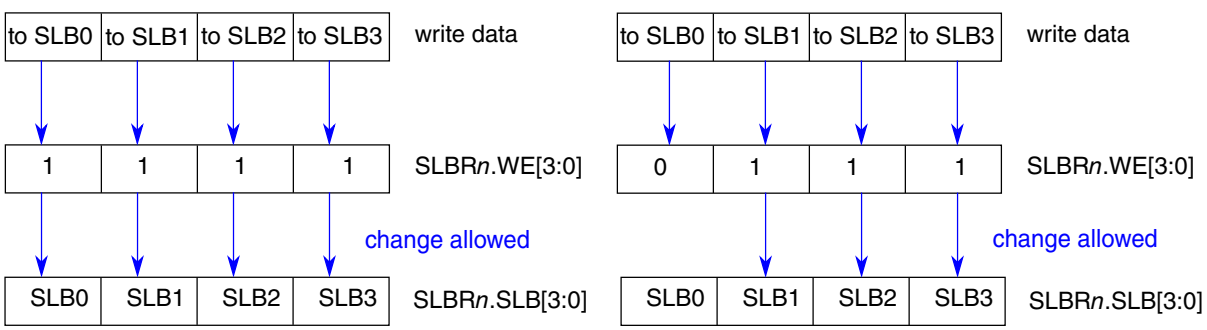


Figure 87-4. Change lock settings directly via Area #4

The previous figure shows four registers that can be protected 8-bit wise. The following figures show registers with 16- and 32-bit protection respectively:

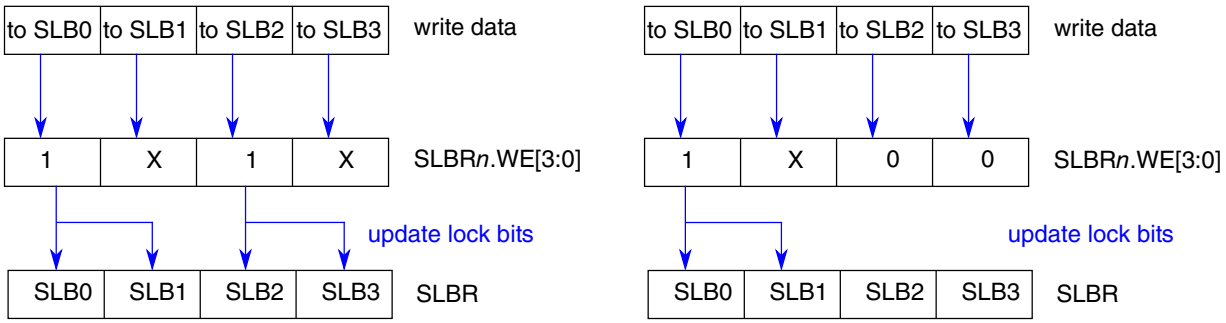


Figure 87-5. Change lock settings for 16-bit protected addresses

On the right side of the previous figure it is shown that the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[1] also. This is done as the address reflected by SLBRn.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBRn.WE[0] must be set while SLBRn.WE[1] does not matter. As the enable bits SLBRn.WE[3:2] are cleared the lock bits SLBRn.SLB[3:2] remain unchanged.

In the example on the left side of the previous figure the data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] and the data written to SLBRn.SLB[2] is mirrored to SLBRn.SLB[3] as for both registers the write enables are set.

The next figure shows a 32-bit wise protected register. When SLBRn.WE[0] is set the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[3:1] also. Otherwise SLBRn.SLB[3:0] remains unchanged.

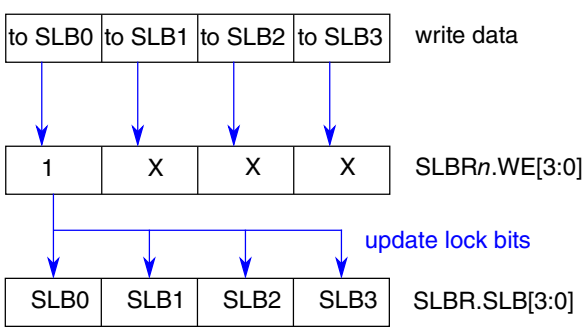


Figure 87-6. Change lock settings for 32-bit protected addresses

The following figure shows a mixed protection size configuration:

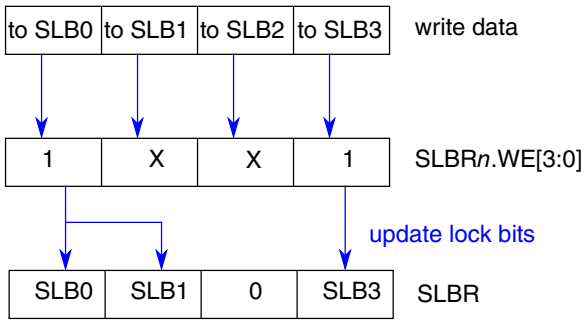


Figure 87-7. Change lock settings for mixed protection

The data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBRn.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBRn.SLB[3] is written to SLBRn.SLB[3].

87.7.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. For example:

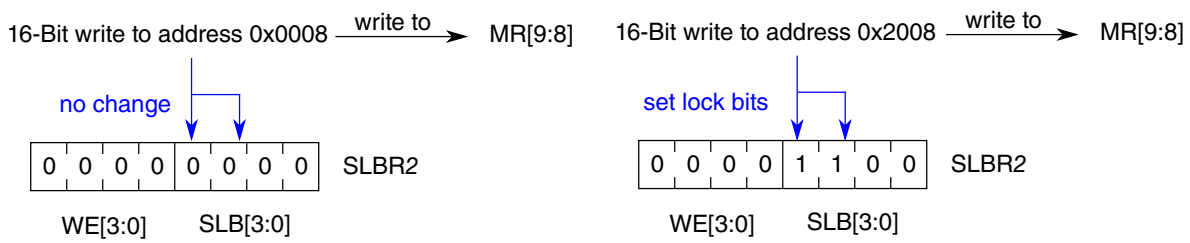


Figure 87-8. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (see the left part of Figure 87-5).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 87-5).

The following figure shows an example where some addresses are protected and some are not:

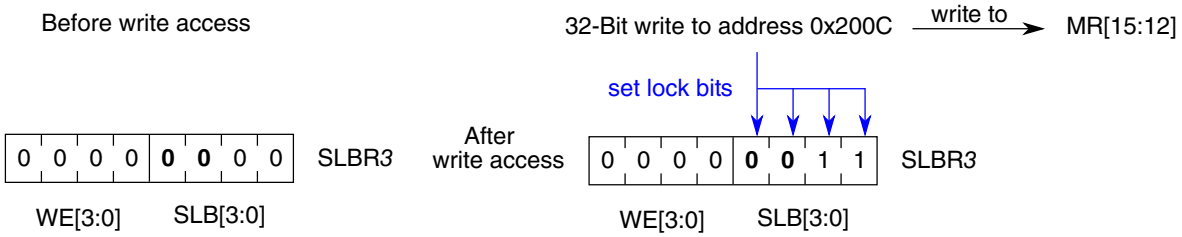


Figure 87-9. Enable locking for protected and unprotected addresses

In the previous figure addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

87.7.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Change lock settings directly via Area #4 and Enable locking via mirror module space (Area #3) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

87.7.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 87-2

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.

6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

87.8 Initialization/application information

The following sections contain information relating to initialization and use of the Register Protection module in applications.

87.8.1 Reset

The reset state of each individual bit is shown within the Register Description section. In summary, after reset, locking for all MRn registers is disabled. The registers can be accessed in Supervisor Mode only.

Chapter 88
Password and Device Security Module (PASS)

88.1 Introduction

88.1.1 Overview

The Password and device security (PASS) module shown in the following figure, receives password challenges and determines their validity. It also maintains the device security and access states.

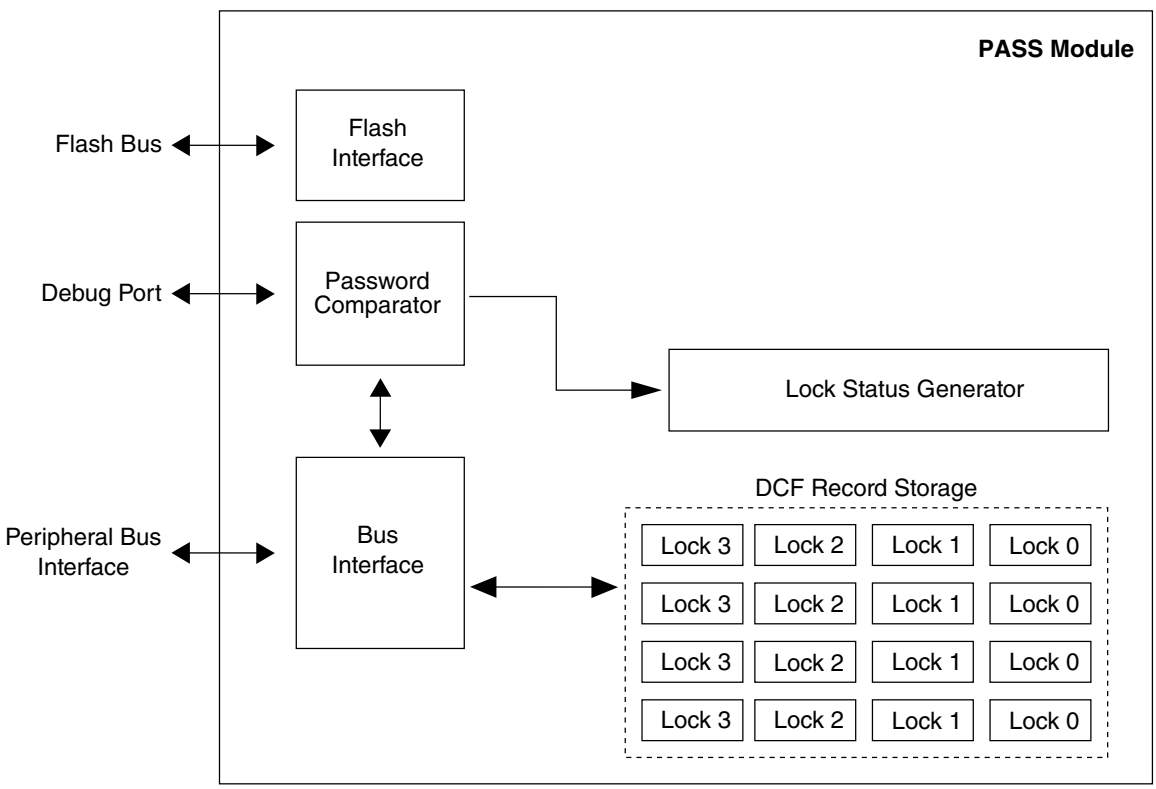


Figure 88-1. PASS module block diagram

88.1.2 Features

The PASS includes these distinctive features:

- Life Cycle status
- Password comparison
- JTAG password comparison
- Production Disable control

88.1.3 Modes of operation

The PASS operates identically in all system modes.

88.2 External signal description

The PASS has 1 external pin that is used for Production Disable control.

88.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers and DCF clients in the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in flash UTEST region.

The following table shows the memory map for the PASS. Note that all addresses are offsets; the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual.

NOTE

Non-implemented registers generate bus aborts if enabled.

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write '0' to these bits; they always read '0'.

PASS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Life Cycle Status Register (PASS_LCSTAT)	32	R	See section	88.3.1/4588
8	Challenge Selector Register (PASS_CHSEL)	32	R/W	0000_0000h	88.3.2/4589
10	Challenge Status Register (PASS_CSTAT)	32	R	0000_0000h	88.3.3/4590
20	Challenge Input Register (PASS_CIN0)	32	W (always reads 0)	0000_0000h	88.3.4/4590
24	Challenge Input Register (PASS_CIN1)	32	W (always reads 0)	0000_0000h	88.3.4/4590
28	Challenge Input Register (PASS_CIN2)	32	W (always reads 0)	0000_0000h	88.3.4/4590
2C	Challenge Input Register (PASS_CIN3)	32	W (always reads 0)	0000_0000h	88.3.4/4590
30	Challenge Input Register (PASS_CIN4)	32	W (always reads 0)	0000_0000h	88.3.4/4590
34	Challenge Input Register (PASS_CIN5)	32	W (always reads 0)	0000_0000h	88.3.4/4590
38	Challenge Input Register (PASS_CIN6)	32	W (always reads 0)	0000_0000h	88.3.4/4590
3C	Challenge Input Register (PASS_CIN7)	32	W (always reads 0)	0000_0000h	88.3.4/4590
D0	Clock Jitter Enable (PASS_CJE)	32	R/W	FFFF_FFFFh	88.3.5/4591
100	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG0)	32	R/W	See section	88.3.6/4592
104	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG0)	32	R/W	See section	88.3.7/4594
108	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG0)	32	R/W	See section	88.3.8/4595
10C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG0)	32	R/W	See section	88.3.9/4596
110	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG1)	32	R/W	See section	88.3.6/4592
114	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG1)	32	R/W	See section	88.3.7/4594
118	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG1)	32	R/W	See section	88.3.8/4595

Table continues on the next page...

PASS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG1)	32	R/W	See section	88.3.9/4596
120	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG2)	32	R/W	See section	88.3.6/4592
124	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG2)	32	R/W	See section	88.3.7/4594
128	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG2)	32	R/W	See section	88.3.8/4595
12C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG2)	32	R/W	See section	88.3.9/4596
130	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG3)	32	R/W	See section	88.3.6/4592
134	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG3)	32	R/W	See section	88.3.7/4594
138	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG3)	32	R/W	See section	88.3.8/4595
13C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG3)	32	R/W	See section	88.3.9/4596

88.3.1 Life Cycle Status Register (PASS_LCSTAT)

The Life Cycle Status Register reflects the production status and DCF Censorship client status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	1	0												
W																
Reset	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LIFE							
W																
Reset	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*

* Notes:
- LIFE field: The reset value derives from a DCF client. If no DCF records have been written, the bit resets to ‘1’.
- CNS field: Reset value depends on the Censorship DCF client

PASS_LCSTAT field descriptions

Field	Description
0 CNS	This field indicates whether the DCF Censorship client has been written with a DCF record to 0x55AA or if the DCF Censorship client contains any other value than 0x55AA. The bit can be set (or cleared) via DCF Record in UTest Flash. **NOTE:** If no DCF record has been written to affect the value of the CNS bit, its reset value will be 1. 1 DCF Censorship Client value is anything other than 0x55AA. 0 DCF Censorship client value is 0x55AA.
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to '1', otherwise it will be 0. (In life cycle FA, the JUN bit will always be 0.) Unsuccessful transmission clears this bit.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 LIFE	This field shows the value of Life Cycle which is decoded by the SSCM module. 1111_110- Customer Delivery 1111_10-- OEM Production 1111_0--- In Field ---0_---- Failure Analysis mode **NOTE:** A dash (-) indicates that the value of the bit is not considered for interpretation of the life cycle.

88.3.2 Challenge Selector Register (PASS_CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When the Master Only (MO) bit of the PASS_LOCK3_PGn register is set, only the master that unlocked access to the passgroup settings will be able to access these registers. The CMST field of the Challenge Status Register will reflect the locking master. When the Master Only (MO) bit of the PASS_LOCK3_PGn register is not set, the passgroup settings can be accessed by any master.

When a master writes this register, its ID sets the CMST field of the Challenge Status register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													GRP		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CHSEL field descriptions

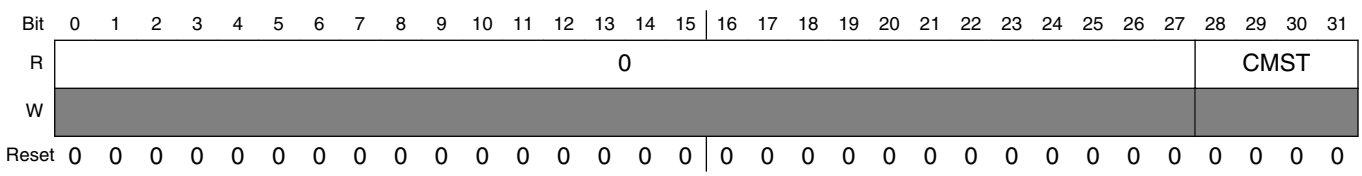
Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 GRP	Password Group to unlock 000 Password Group 0 001 Password Group 1 010 Password Group 2 011 Password Group 3 100 Reserved 101 Reserved 110 Reserved 111 Reserved

88.3.3 Challenge Status Register (PASS_CSTAT)

This register provides status information for the password challenge.

Only the master shown in this register can access the CINn registers - access by other masters result in a comparison fail regardless of the values written.

Address: 0h base + 10h offset = 10h



PASS_CSTAT field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CMST	ID of the master which has last written the CHSEL register

88.3.4 Challenge Input Register (PASS_CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7 in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed. In case of a successful password comparison, the PASS module stores the ID of the master which has written CIN0 to CIN7 in the PASS_LOCK3_PGn register of the selected password group, as well as clears the PGL bit of that register.

All accesses to these registers need to be made by the bus master that wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

Address: 0h base + 20h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	PW32																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CIN*n* field descriptions

Field	Description
0–31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

88.3.5 Clock Jitter Enable (PASS_CJE)

Software accessible one time writable register. This register stores the jitter enable status for Pseudo Random Clock Divider Module (PRCD). The default value of this register is '1'. When this bit is cleared, the PRCD is enabled and the baud rate clock to several serial comms modules include large amounts of psuedo random clock edge jitter, disabling the serial comms modules. For details about which serial comms modules are affected, see the clocking section of the Reference Manual.

For more information on Production disable, see Production disable

Debug access is blocked when Production disable bit[0] (Gated Debug Enable) is asserted, the Chip is In Field and Clock Jitter is not enabled.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1															CJE
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PASS_CJE field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
31 CJE	Clock Jitter Enable. 0 Clock Jitter is enabled. The baud rate clock to Serial Comms modules includes large amounts of pseudo random clock edge jitter 1 Clock Jitter is disabled. The baud rate clocks function normally

88.3.6 Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the life cycle has matured to *OEM Production* or older.

LOCK0_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is '1', indicating that a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes to this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

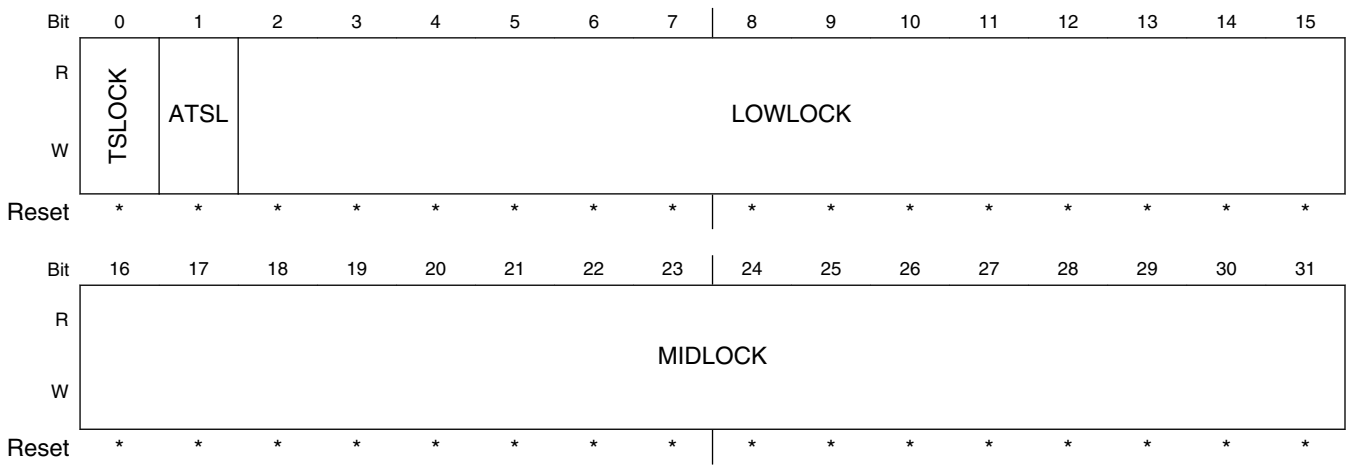
Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

When Production Disable register bit[1](BAF Plock Hardware Interlock enable) is asserted and life cycle is In Field, then the external signal is OR'ed with BAF_plock(low_lock[0]) of LOCK0_PGx.

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d



* Notes:

- MIDLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.
- LOWLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.
- ATSL field: If no DCF records have been written to change the ATSL bit state, the bit will reset to ‘1’.
- TSLOCK field: If no DCF records have been written to change the TS bit state, the bit will reset to ‘1’.

PASS_LOCK0_PG*n* field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable).
1 ATSL	Alternate Interface UTest NVM Lock. This bit is used to lock the UTest NVM block from programs on the alternate interface (erase protection not needed since UTest NVM is OTP and not erasable).
2–15 LOWLOCK	Low Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16–31 MIDLOCK	Mid Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

88.3.7 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK1_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset . The default value before any DCF record is written is '1', which indicates a block is locked. If no DCF records have been written to change the lock state, the bits will reset to '1'.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group to prevent permanent and irreversible locking of Flash blocks.

CAUTION

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- HIGHLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.

PASS_LOCK1_PG*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

88.3.8 Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle is matured to OEM Production or older.

NOTE

Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in this section.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is ‘1’, which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L_256LCK																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- L_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit resets to ‘1’.

PASS_LOCK2_PG*n* field descriptions

Field	Description
0–31 L_256LCK	Lower 256KByte Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

88.3.9 Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK3_PGn register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Once a censored device enters the Failure Analysis life cycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash Utest region controlled by LOCK3[0] is read unlocked in Failure Analysis life cycle.

NOTE

Do not toggle DBL and RLx bits with the same write command. Use one instruction to program the DBL bit and a separate instruction to toggle the RLx bits.

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0			RL4	RL3	RL2	RL1	RL0
W																
Reset	*	*	*	0	1	1	1	1	0	0	0	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	U_256LCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- U_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- RL0 field: If no DCF records have been written to change the RL0 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL1 field: If no DCF records have been written to change the RL1 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.

- RL2 field: If no DCF records have been written to change the RL2 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL3 field: If no DCF records have been written to change the RL3 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL4 field: If no DCF records have been written to change the RL4 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- MO field: If no DCF records have been written to change the MO bit lock state, the bit will reset to '1'.
- DBL field: If no DCF records have been written to change the DBL bit lock state, the bit will reset to '1'.
- PGL field: Reset value depends on life cycle. If life cycle is Customer Delivery or earlier, the value will be 0, otherwise 1.

PASS_LOCK3_PG*n* field descriptions

Field	Description
0 PGL	Password Group Lock. This bit may be set in software to lock the password group, when the device is older than Customer Delivery. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF load from Flash. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If life cycle is Customer Delivery or earlier, this bit has no effect and the registers are always unlocked.)
1 DBL	Debug Interface Lock. The default value of this bit would be '1' if not updated from flash. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed (Except HSM) 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked
2 MO	Master Only. 0 All masters can modify the passgroup settings if PGL is cleared 1 Only the Master which unlocked the passgroup can change the passgroup settings
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MSTR	Master Access. Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 RL4	Read Lock Regison 4 lock bit. The default value of this bit would be '1' if not updated from flash.

Table continues on the next page...

PASS_LOCK3_PG*n* field descriptions (continued)

Field	Description
	0 Region 4 is not protected. 1 Region 4 is protected.
12 RL3	Region 3 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 3 is not protected. 1 Region 3 is protected.
13 RL2	Region 2 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 2 is not protected. 1 Region 2 is protected.
14 RL1	Region 1 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 1 is not protected. 1 Region 1 is protected.
15 RL0	Region 0 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 0 is not protected. 1 Region 0 is protected.
16–31 U_256LCK	Upper 256 KB Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

88.4 DCF clients

The following table shows the DCF clients for the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual.

Table 88-1. PASS DCF clients

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	Censorship
Production Disable					

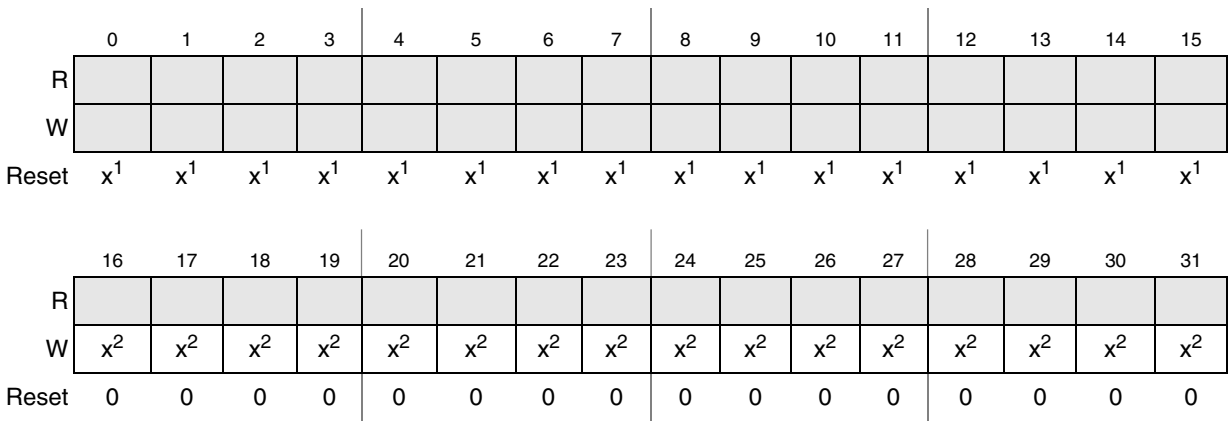
Table continues on the next page...

Table 88-1. PASS DCF clients (continued)

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section
None	Production Disable	None	0x0000_0000	0x00C0	Production Disable

88.4.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region.



1 This bit is not defined

2 This value is loaded by DCF Record during reset

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other than 0x55AA, the MCU is censored

88.4.2 Production Disable

Production Disable is a DCF client that is not accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region. The Production disable DCF client defines the Debug Disable and Hardware Interlock aspects of the Production Disable feature.

The DCF client implements 2 bits with a default value of ‘0’. The DCF client can only be written once and writes from subsequent DCF records are ignored.

For more information on Production disable, see Production disable

DCF Client Address: 0x00C0 Production Disable — Access: DCF Record Write Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W															HI[1]	DD[1]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[1] This value is loaded by DCF Record during reset

Figure 88-2. Production Disable DCF client

Table 88-2. Production Disable DCF Client field descriptions

Field	Description
0-29	This bit is reserved and always has the value 0.
30 HI	Hardware Interlock '0' = BAF flash block programming is controlled by a bit in PASS_LOCK0_PG[LOWLOCK] that controls the BAF (see chip-specific PASS information). '1' = BAF flash block is locked for programming while external hardware pin is asserted '0'
31 DD	Debug disable. '0' = Debug Interface is Enabled and Disabled using Censorship, JTAG password and PASS_LOCK3[DBL] bit (see chip-specific PASS information). '1' = Debug Interface is disabled while Clock Jitter is NOT enabled

88.4.2.1 Debug Disable (DD)

Debug Disable is an optional feature that, once the life cycle is In Field or beyond, overrides all other debug controls and disables the debug interface. The Debug Disable feature has a higher priority than Censorship, JTAG password and the PASS LOCK3[DBL] bit. All Debug access is blocked while Clock Jitter is *not* enabled.

Once Clock Jitter *is* enabled, the Debug interface is *not* disabled and can be enabled temporarily or permanently using Censorship, JTAG password or the PASS_LOCK3[DBL] bit.

88.4.2.2 Hardware Interlock (HI)

Hardware Interlock is an optional feature that, once the life cycle is In Field or beyond, prevents programming of the BAF Flash block while an external hardware pin is asserted low. This feature has higher priority than PASS_LOCK0[BAF] bits. While the pin is asserted, no software action anywhere in the MCU can cause the BAF to be programmed.

Once the hardware pin is asserted high, programming of BAF is controlled by PASS_LOCK0[BAF] bits.

88.5 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

88.5.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access and flash memory read protection.

The Debug Interface access is controlled by the CNS and LIFE bits of the LCSTAT register according to the Debug Interface Access truth table. The flash memory read protection is also controlled by the CNS and LIFE bits of the LCSTAT register according to the Flash memory read protection truth table.

The CNS bit is set or cleared by executing a DCF record that writes to the Censorship DCF client (See Censorship). The LIFE bits of the LCSTAT register are controlled by the SSCM using user programmed values from specific locations in the UTEST flash.

88.5.2 Debug Interface Access

The debug interface is fully enabled in Customer Delivery life cycle. But from OEM Production life cycle, it can be blocked using censorship and unlocked by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 registers (See Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)). Both these debug enabling options are further conditioned by life cycle and Production Disable.

The table below describes all options for enabling/blocking Debug Interface access

Table 88-3. Debug Interface Enable Truth Table

Input Conditions						Outcome
Lifecycle	**Production Disable mode**	**Clock Jitter Enable**	**Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
Customer Delivery	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Enabled
OEM Production	Don't Care	Don't Care	Not censored	Don't Care	Don't Care	Enabled
			Censored	Unlocked (DBL=0)	Don't Care	Enabled
				Locked (DBL=1)	Matched	Enabled
					Not Matched	Blocked
In Field	Off	Don't Care	Not censored	Don't Care	Don't Care	Enabled
			Censored	Unlocked (DBL=0)	Don't Care	Enabled
				Locked (DBL=1)	Matched	Enabled
					Not Matched	Blocked
	On	Jitter Enabled	Not censored	Don't Care	Don't Care	Enabled
			Censored	Unlocked (DBL=0)	Don't Care	Enabled
				Locked (DBL=1)	Matched	Enabled
					Not Matched	Blocked
		Jitter disabled	Don't Care	Don't Care	Don't Care	Blocked
Failure Analysis	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Enabled

88.5.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, can be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled until the next destructive reset.

88.5.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Life Cycle, Censorship and whether the Debug Interface access is enabled or blocked. The Flash memory read protection truth table describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The flash memory is also read protected when the life cycle status is Failure Analysis. This life cycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (3 to 5 depending on the device). The assignment of the regions is defined in the device reference manual.

Table 88-4. Flash Memory Read Protection Truth Table

INPUTS				OUTPUTS	
Life Cycle	**Censorship**	**Debug Interface Enable from HSM**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[4:1] (Other Regions)**
Customer Delivery	Don't Care	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Blocked	Don't Care	Readable	Readable
		Enabled	Locked (1)	Read Blocked	Read Blocked
			Unlocked (0)	Readable	Readable
Failure Analysis	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Don't Care	Locked (1)	Readable	Read Blocked
			Unlocked (0)	Readable	Readable

88.5.4 Production Disable

88.5.4.1 Overview and rationale

Production Disable is an optional feature that, once enabled, allows the MCU to function completely normally without any constraints, except the Debug interface is disabled.

Once the customer has decided that the Debug Interface should be re-enabled, the process to re-enable it makes the MCU unusable in a production environment by adding clock jitter to the baud rate clocks of several serial communications modules (normally CAN and FlexRay). This jitter means those serial communications modules are not able to communicate on the network.

As this Production Disable feature renders the MCU unusable in the application, inadvertent activation would be disastrous. Therefore a feature is included that prevents this Jitter activation unless an external hardware pin is asserted.

88.5.4.2 Enabling

The feature is enabled via the Production Disable DCF client in the PASS module. Bit 0 enables the Debug Interface Disable. Bit 1 enables the Hardware Interlock pin. See Production Disable for details.

This information can be written by adding a DCF Record to the DCF record list in Flash UTEST. The DCF Client is configured to accept only one write. Subsequent DCF records that write to this DCF client are ignored.

If no DCF Record writes to this client, the default state of both features is disabled. If, for a particular application, there is no intention of ever using this feature, it would be good practice to add a DCF Record that writes this client to 0x0000_0000. This is not essential, but ensures the feature can not be enabled accidentally.

88.5.4.3 Activation

The BAF code is always the first code executed out of reset. As part of the normal boot sequence, The BAF code copies the ‘Clock Jitter Activation’ constant value from a location in the BAF code space, to the Clock Jitter Register in the PASS module (See Clock Jitter Enable (PASS_CJE) for details).

This register is write once. The default value for the Clock Jitter Activation constant in BAF Flash is 0xFFFF_FFFF (erased state)

To enable the Clock Jitter feature, the value of the Clock Jitter Activation constant in BAF Flash should be programmed to make the least significant bit ‘0’. After the next reset, the BAF copies this updated value to the Clock Jitter Register and the feature is activated.

The BAF Flash block is assigned as OTP and it is protected from programming by a bit in the PASS LOCK0 registers; but the BAF Flash block can further be protected from programming by an external hardware pin. If the Hardware Interlock feature is enabled, while this pin is logic ‘0’ the BAF Flash block can not be programmed.

88.5.4.4 Clock jitter Truth table

The following table shows the relationship between Life Cycle, Production Disable & Clock Jitter enable input conditions versus the state of the Baud Rate Clock.

Table 88-5. Clock Jitter Truth Table

Life Cycle	Production Disable[Debug Disable]	Clock Jitter Enable	Baud Rate Clock State
JDP Production Customer Delivery OEM Production In Field	Don't Care	0x0001	Normal - No Jitter
		0x0000	Jitter
Failure Analysis	Off [0]	0x0001	Normal - No Jitter
		0x0000	Jitter
	On[1]	Don't Care	Jitter

88.6 Initialization/application information

88.6.1 Reset

The reset state of each individual bit is shown within the Register Description section. However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

88.6.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 88-6. CS/ADDR Values for Lock Registers

Register	DCF Command Word
PW Group 0	
Lock0	0xxxxx[1] _0100
Lock1	0xxxxx[1] _0104
Lock2	0xxxxx[1] _0108
Lock3	0xxxxx[1] _010C
PW Group 1	
Lock0	0xxxxx[1] _0110
Lock1	0xxxxx[1] _0114
Lock2	0xxxxx[1] _0118
Lock3	0xxxxx[1] _011C
PW Group 2	
Lock0	0xxxxx[1] _0120
Lock1	0xxxxx[1] _0124
Lock2	0xxxxx[1] _0128
Lock3	0xxxxx[1] _012C
PW Group3	
Lock0	0xxxxx[1] _0130
Lock1	0xxxxx[1] _0134
Lock2	0xxxxx[1] _0138
Lock3	0xxxxx[1] _013C

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See chip specific PASS information for the Client Select assignment

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in Table 88-7.

Table 88-7. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00FF	PW Group 1, Lock0
0x0C	0xxxxx[1]_0110	Set MIDLOCK to 0xFF
0x10	0xFFFF_FFFF	Non-Programmed Addresses
0x14	0xFFFF_FFFF	
…	…	

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See Device reference manual for the Client Select assignment

Chapter 89
Tamper Detection Module (TDM)

89.1 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that requires software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prevent* an erase operation on any block within a TDR—it requires a record to be written before the operation can execute. Collectively, the records are referred to as a "diary". At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

The following figure shows the block diagram of the TDM and its interface to other system components.

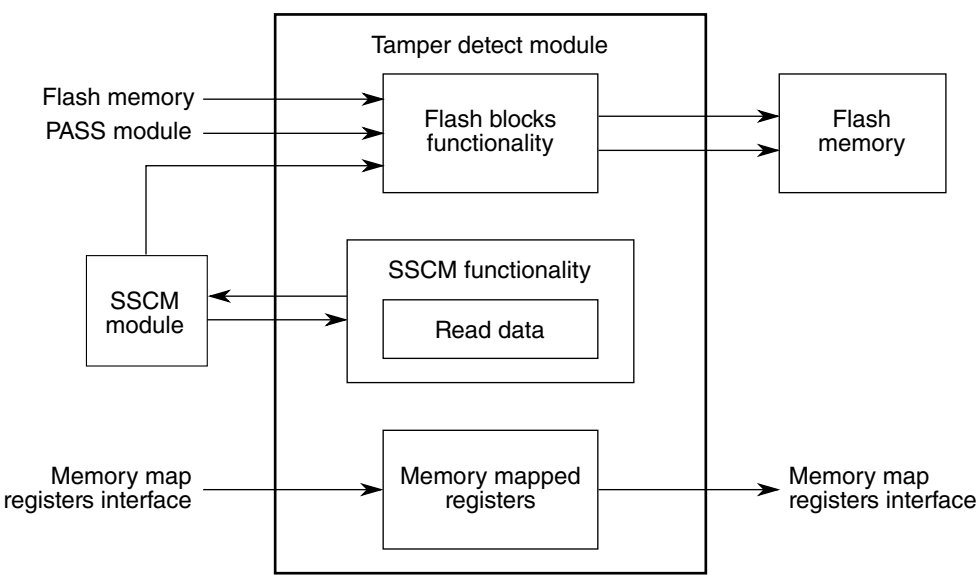


Figure 89-1. Tamper Detection Module block diagram

Security information is passed to the TDM by the SSCM module. Among the data are:

- OTP (One Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- block assignments for the TDRs
- Tamper region override information (via the DCF bus)
- Software tamper region override

The TDM module controls the flash blocks for erase operations and sets them as OTP.

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

NOTE

The pass module provides the control for erase operation for each flash block as well as for each region of flash memory, and the PASS settings have priority over the settings of TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

89.2 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 6 parts (TDRs), each consisting of 256 64-bit records.
  - The diary block should be assigned as OTP region in flash.
  - The diary area can store any type of relevant information.
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled.
- 32-bit Erase Lock DCF clients for each flash memory TDR defined.
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records[1]
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation.
- Memory mapped registers interface for reading via software of specific registers.

1. The Tamper Region Override DCF record is a "write one only" DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

89.3 External signal description

There are no external signals driving or being driven by the TDM.

89.4 DCF clients

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/ erase activity
- Permanently disable one or more Tamper Regions
- Permanently disable ability for software to permanently override one or more TDRs
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

Table 89-1. DCF clients

DCF Client Name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
Software Tamper Region Override Disable	Stores information to disable the software mechanism to override the TDR	write once, write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR. • n specifies a TDR • m specifies the flash blocks as per the LOCK registers of the flash memory.	write 0 to 1 only

89.4.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. The following diagram depicts this client.

CAUTION

- This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.
- This DCF client is writable from 1 to 0 only.

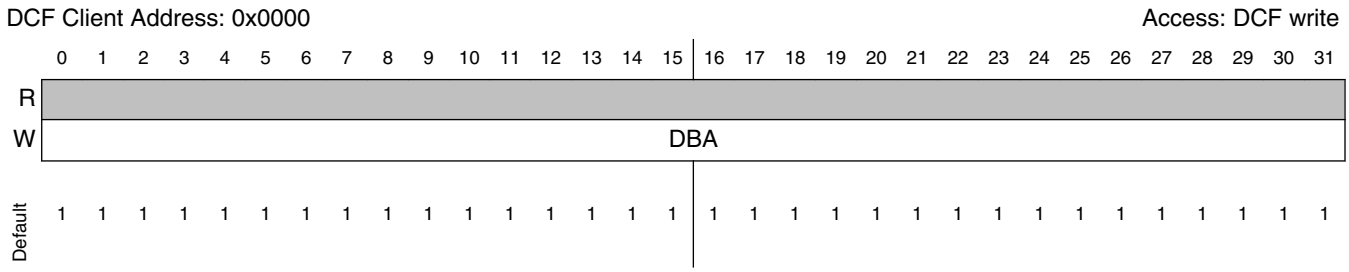


Table 89-2. Diary Base Address (DBA) DCF client field description

Field	Description
0–31 DBA	Diary Base Address. This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. **NOTE:** Diary Base Address must be on a 16 KB boundary.

89.4.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. The following figure shows the diagram for this register.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-3. Tamper Region Override (TO) DCF client field descriptions

Field	Description
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the TOE fields.
26–31 TOE5 to TOE0	TDR Override Enable. Each TOE*n* field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written. 1 Diary override. The blocks assigned to this TDR can be erased WITHOUT writing to the diary.

89.4.3 Software Tamper Region Override Disable (STO_DIS) DCF client

This DCF client holds information to disable the software mechanism to override the tamper detect regions.

CAUTION

- This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.
- This DCF client is writable from 0 to 1 only.

DCF Client Address: 0x0002 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STO_DIS5	STO_DIS4	STO_DIS3	STO_DIS2	STO_DIS1	STO_DIS0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-4. Software Tamper Region Override Disable (STO_DIS) DCF client field descriptions

Field	Description
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the STO_DIS fields.
26–31 STO_DIS5 to STO_DIS0	Software TDR Override Disable TDR*n*. Each STO_DIS*n* field, where *n* represents the TDR number, holds information to disable the software mechanism to override TDR*n*. 0 Software tamper detect override mechanism supported 1 Software tamper detect override mechanism disabled. Any attempt to override TDR*n* by loading a service key value in STO_KEY*n* register is ignored.

89.4.4 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. The following diagram depicts this client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0008 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	LOWOTPEN[11:0]												MIDOTPEN[15:0]															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-5. One Time Programmable Enable (OTPEN0) DCF client

Field	Description
0–3 Reserved	This field is reserved and should be written with all zeroes when writing to this client.
4–15 LOWOTPEN	Low Block OTP Enable. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
16–31 MIDOTPEN	Mid Block OTP Enable. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

89.4.5 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0009 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHOTPEN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-6. One Time Programmable Enable (OTPEN1) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHOTPEN	High Block OTP Enable. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

89.4.6 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000A Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	256KBOTPEN																															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-7. One Time Programmable Enable (OTPEN2) DCF client field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

89.4.7 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK3 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000B — Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256KBOTPEN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-8. One Time Programmable Enable (OTPEN3) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in the upper range of 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

89.4.8 Tamper Region Assignment DCF client (TDRn_LOCK0)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK0 register in the flash memory. The following diagram depicts the TDRn_LOCK0 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 0) Access: DCF write

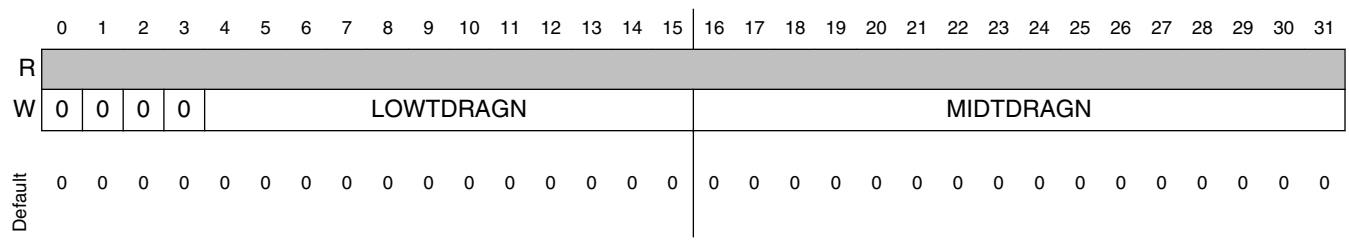


Table 89-9. Tamper Region Assignment DCF client (TDRn_LOCK0)

Field	Description
0–3 Reserved	This field is reserved and should be written with zeroes when writing to this client.
4–15 LOWTDRAGN	Low Block Tamper Region Assignment. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn
16–31 MIDTDRAGN	Mid Block Tamper Region Assignment. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

89.4.9 Tamper Region Assignment DCF client (TDRn_LOCK1)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK1 register in the flash memory. The following diagram depicts the TDRn_LOCK1 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 1) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHTDRAGN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-10. Tamper Region Assignment DCF client (TDRn _LOCK1)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHTDRAGN	High Block Tamper Region Assignment. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

89.4.10 Tamper Region Assignment DCF client (TDRn_LOCK2)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK2 register in the flash memory. The following diagram depicts the TDRn_LOCK2 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 2) Access: DCF write

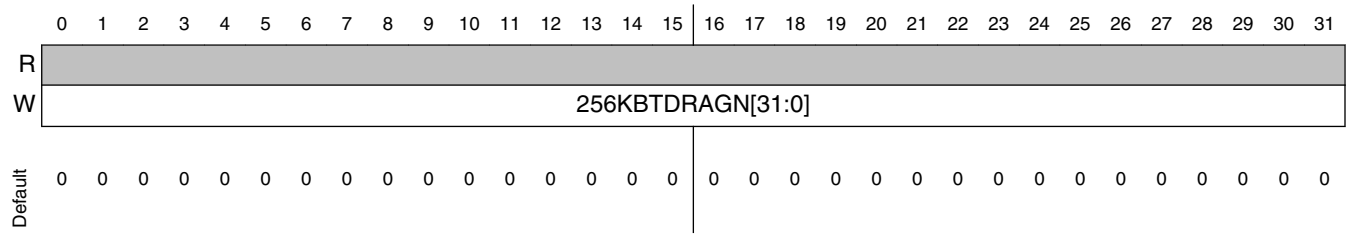


Table 89-11. Tamper Region Assignment DCF client (TDRn_LOCK2)

Field	Description
0–31 256KBTDRAGN[31:0]	256 KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

89.4.11 Tamper Region Assignment DCF client (TDRn_LOCK3)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK3 register in the flash memory. The following diagram depicts the TDRn_LOCK3 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 3) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256KBTDRAGN[47:32]															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 89-12. Tamper Region Assignment DCF client (TDRn_LOCK3)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBTDRAGN[47:32]	256KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

89.5 Memory Map and Registers

NOTE

The Tamper Detection Module's memory-mapped registers are read-only. Any attempted write generates a transfer error.

TDM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	TDR Status Register (TDM_TDRSR)	32	R	See section	89.5.1/4622
4	Last Flash Programmed Address Register (TDM_LFPAR)	32	R	0000_0000h	89.5.2/4624
8	Diary Base Address (TDM_DBA)	32	R	See section	89.5.3/4625
10	Software Tamper Override Key Region (TDM_STO_KEY0)	32	R/W	0000_0000h	89.5.4/4625
14	Software Tamper Override Key Region (TDM_STO_KEY1)	32	R/W	0000_0000h	89.5.4/4625

Table continues on the next page...

TDM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18	Software Tamper Override Key Region (TDM_STO_KEY2)	32	R/W	0000_0000h	89.5.4/4625
1C	Software Tamper Override Key Region (TDM_STO_KEY3)	32	R/W	0000_0000h	89.5.4/4625
20	Software Tamper Override Key Region (TDM_STO_KEY4)	32	R/W	0000_0000h	89.5.4/4625
24	Software Tamper Override Key Region (TDM_STO_KEY5)	32	R/W	0000_0000h	89.5.4/4625

89.5.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, i.e., locked or unlocked, of each tamper region, i.e., TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

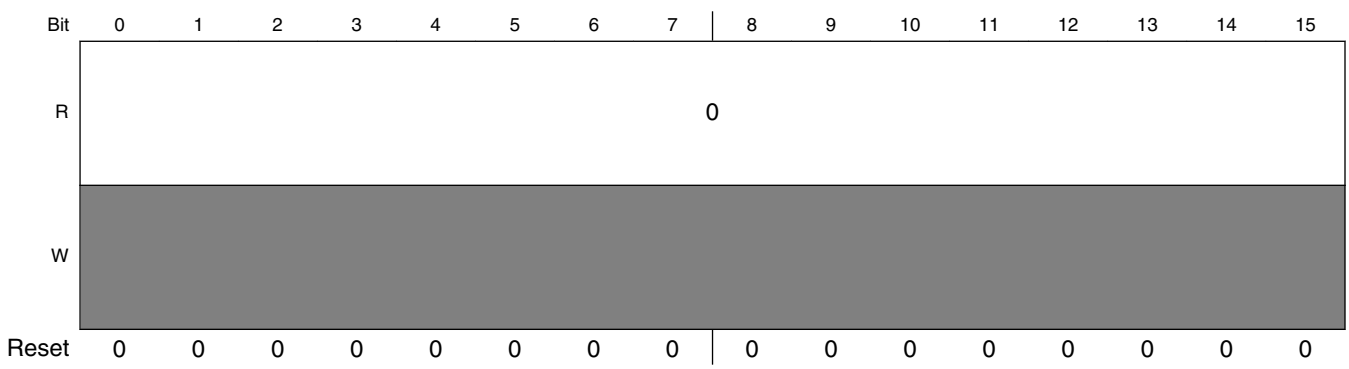
The register contains a status bit for each TDR. Status is determined as follows:

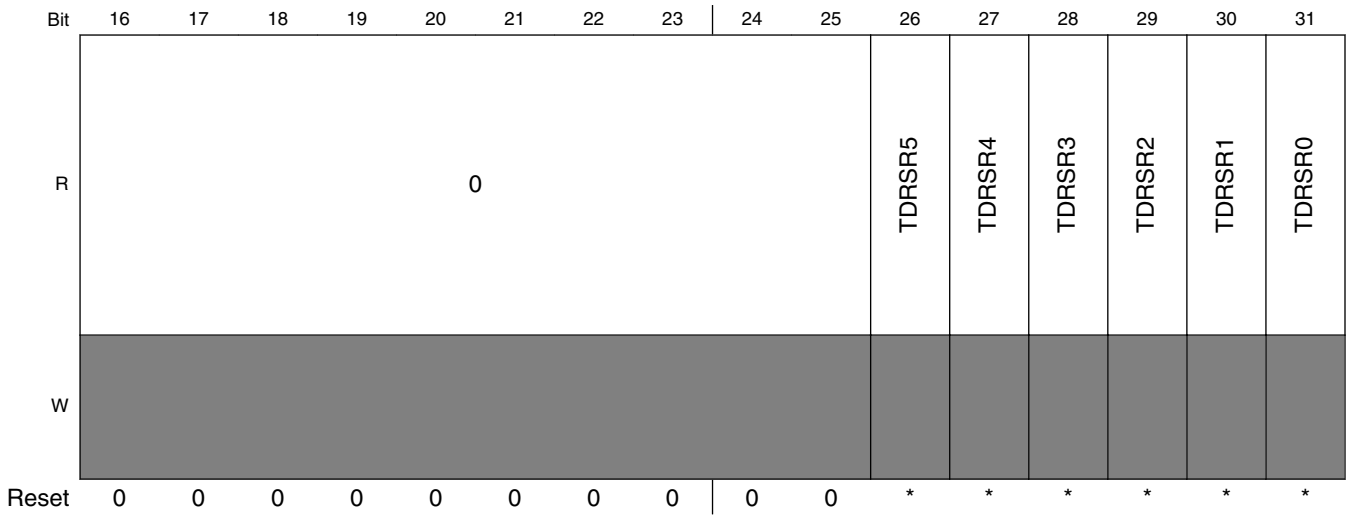
1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

NOTE

1. TDRSR is a read-only register and writes have no effect. Write operations result in a transfer error.
2. If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Address: 0h base + 0h offset = 0h





* Notes:
- TDRSR0 field: See register description.
- TDRSR1 field: See register description.
- TDRSR2 field: See register description.
- TDRSR3 field: See register description.
- TDRSR4 field: See register description.
- TDRSR5 field: See register description.

TDM_TDRSR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
27 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status.

Table continues on the next page...

TDM_TDRSR field descriptions (continued)

Field	Description
	0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

89.5.2 Last Flash Programmed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LFPAR																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_LFPAR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LFPAR	Last Flash Programmed Address. These bits represent the address of the last successful programming operation.

89.5.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBA																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value of this register is set via DCF record.x = Undefined at reset.

TDM_DBA field descriptions

Field	Description
0–31 DBA	Diary Base Address. These bits represent the address of the diary location in the flash. **NOTE:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

89.5.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region n, where n denotes the Tamper Detect Region number.

Caution

1. The STO_KEYn registers can only be written once.
2. The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Address: 0h base + 10h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	STO_KEY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_STO_KEY*n* field descriptions

Field	Description
0–31 STO_KEY	Software Tamper Override Key Diary n. This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Caution:** The STO_KEYn registers can only be written once.

TDM_STO_KEY*n* field descriptions (continued)

Field	Description

89.6 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

89.6.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the diary is defined by a DCF Record.

The diary is divided into 6 parts where each part consists of 256 × 64-bit records, to record up to 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 x 32-bit internal hardware registers that are initialized by DCF records that define the tamper region. These registers mirror exactly the LOCK registers bits from the flash module.

Each TDRn_LOCK*m* register is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR. As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensure that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double word program operation, the flash module indicates to the TDM the address of the programmed double word. TDM determines whether the programmed data was written to the correct TDR diary Location and then unlocks the

blocks for erase operation for the corresponding TDR. TDM ensures that this erase operation on unlocked blocks is allowed until the completion of the erase sequence, with program completion interrupt enabled (PECIE) or till next reset.

NOTE

All blocks associated to a TDR where an erase operation is allowed can be erased all together.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

NOTE

There is no restriction on the type or format of data that can be programmed in the diary, so software must verify whether the diary is full. If a TDR's diary area is full, the tamper detect mechanism can be disabled for that TDR by creating a Tamper Region Override (TO) DCF record with the TDR's corresponding TDR Override Enable (TOE) bit set to '1'. This allows erase operations to be performed to the TDR without any diary entry. Alternatively, if another TDR is available it can be defined to contain the same blocks to extend the diary length for that TDR.

The setting from “override tamper region lock” DCF record (Diary) has priority over the settings in the TDRn_LOCK0- 3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, the settings from PASS_LOCK*x*_PG*n* registers have priority over TDR*n*_LOCK*m* settings. For example, if a block is password-protected for write operations, the TDR lock setting is ignored.

NOTE

The settings in the Tamper Region Override Register have priority over the TDR*n*_LOCK*m* registers. The PASS module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via TDM, but the same blocks are programmed to be locked for erase operation in PASS, the blocks end up being locked for erase operation.

89.6.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 x 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as TDR*n*_LOCK*m* where *n* is the number of a tamper region, that is, 0- 5 and *m* varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- '1' means that the corresponding flash block is assigned to this TDR
- '0' means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They can not be read nor written. They can only be initialized by DCF record within the UTest flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

NOTE

No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

89.6.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming operation is 8 bytes to honor OTP restrictions. Therefore, each diary can hold a maximum of 256 64-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core and from the HSM.

NOTE

To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once and the default bit state is 1.

The DCF clients can not be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

89.6.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes. Therefore, each diary region limits the erases of the assigned blocks to 256 erases.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once and the default bit state is ‘1’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 6 TDRs, 12 KB of diary space is required.

The diary base address must be at a 16 KB boundary. Therefore the least significant 14 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xx00_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

89.6.5 TDR lock status

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as "TDR Erase Blocked". See TDR Status Register (TDM_TDRSR) for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning "TDR Erase Enabled".

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. See TDR Status Register (TDM_TDRSR). The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. See Last Flash Programmed Address Register (TDM_LFPAR).

89.6.6 TDR override

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to ‘0’ in each 64 bit double word within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. See the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

The application can also override the diary by loading a service key to the STO_KEY*n* program-visible register, where n denotes the tamper detect region. By loading the value 0x55AA5A5A into STO_KEYn register, the corresponding tamper detect region is overridden. This software override mechanism is disabled if the STO_DISn field of the STO_DIS DCF Client is set.

89.6.7 One time programmable (OTP)

89.6.7.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next reset.

Chapter 90
Wakeup Unit WKPU

90.1 Introduction

The Wakeup Unit (WKPU) supports one external source that can cause non-maskable interrupts to on-chip cores and wakeup events to the system. This NMI external source can also generate a functional reset event to the MC_RGM. The following figure is a block diagram of the WKPU and its interfaces to other system components.

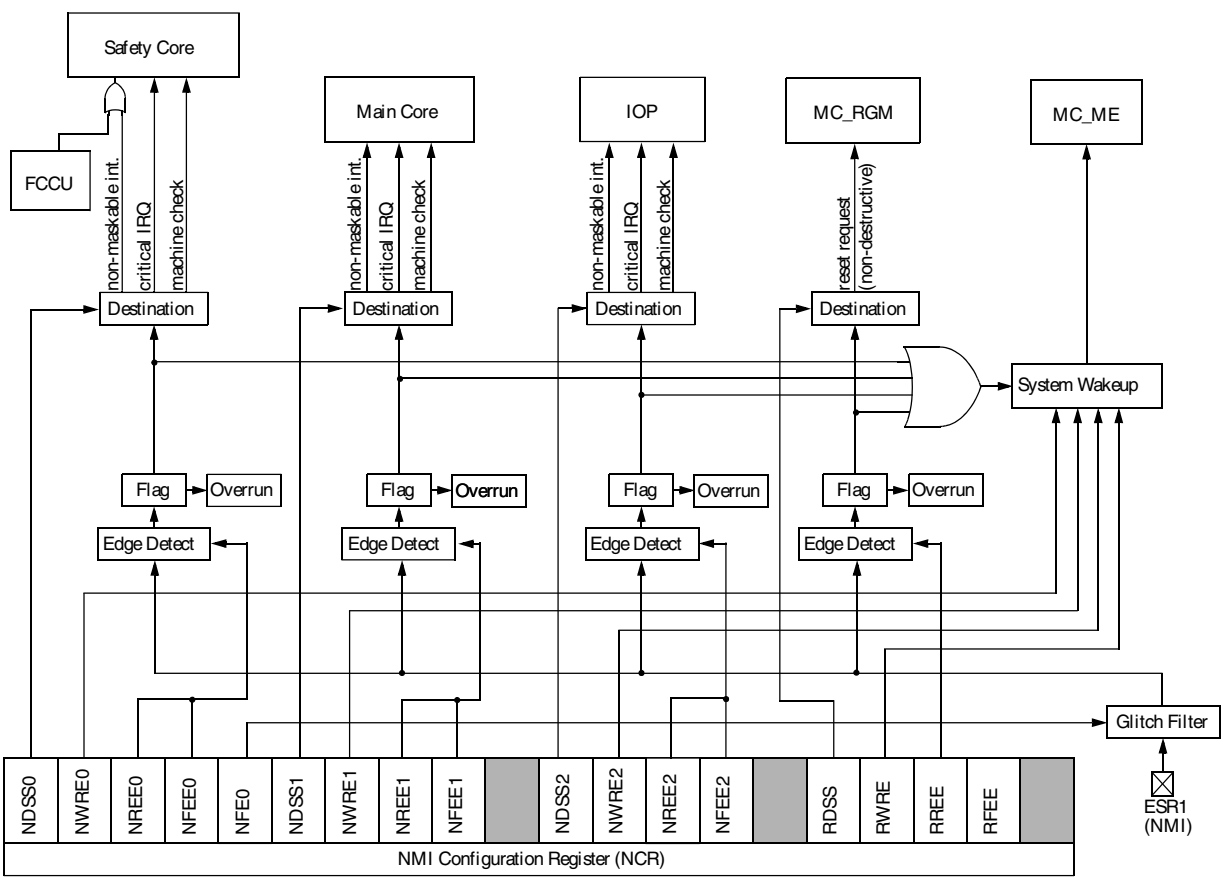


Figure 90-1. WKPU block diagram

90.1.1 Features

The WKPU supports these features:

- One external NMI source
- One analog glitch filter
- Independent interrupt destination for each core (safety core, main core, and IOP):
  - Non-maskable interrupt
  - Critical interrupt
  - Machine check request
- Active edge selection control for events to each core and functional reset event to MC_RGM
- Configurable wakeup event from low-power mode to the MC_ME
- Configurable wakeup event to exit from Stop mode to the MC_ME
- WKPU register configuration lock

90.2 External signal description

There is a single, active-edge selectable, NMI pin on the device that connects to the WKPU block. See this chip's signal description information for details on this pin.

90.3 WKPU memory map and registers

This section provides a detailed description of all registers accessible in the WKPU module.

NOTE

Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

WKPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	NMI Status Flag Register (WKPU_NSR)	32	R/W	0000_0000h	90.3.1/4635
8	NMI Configuration Register (WKPU_NCR)	32	R/W	0000_0060h	90.3.2/4637

90.3.1 NMI Status Flag Register (WKPU_NSR)

This register holds the status flags of the NMI external source pin and reset request. This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI or reset request event is detected. The overrun flag is set whenever an NMI or reset request event is detected and the status flag is set (i.e., has not yet been cleared). The reset request overrun is a don't-care since the device will be reset on the first event.

The status flags are set independently of the NDSS/RDSS bits. Therefore, these flags can set but not generate a non-maskable interrupt or system wakeup if NWRE/RWRE = 0 or if NDSS is set to "no request".

NOTE

The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

NOTE

NIF is set when all the following are true:

- NMI Destination Source Select is disabled (NDSS = 11)
- NMI System Wakeup Request is disabled (NWRE = 0)
- An NMI input event is seen

Also see the notes in NMI Configuration Register (WKPU_NCR).

Address: FEED_0000h base + 0h offset = FEED_0000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NIF0	NOVF0	0						NIF1	NOVF1	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIF2	NOVF2	0						RIF	ROVF	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_NSR field descriptions

Field	Description
0 NIF0	NMI Status Flag 0 (safety core) If enabled (i.e., NCR[NREE0] or NCR[NFEE0] is set), an NMI input event will set NIF0. NIF0 causes an interrupt request based on the NMI destination, as configured by NCR[NDSS0], and/or a system wakeup request based on NCR[NWRE0]. 0 No event has occurred on the pad 1 An event as defined by NCR[NREE0] and NCR[NFEE0] has occurred
1 NOVF0	NMI Overrun Status Flag 0 (safety core) This field is a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. NOVF0 causes an interrupt request based on the NMI destination, as configured by NCR[NDSS0], and/or a system wakeup request based on the NCR[NWRE0] bit configuration. 0 No overrun has occurred on NMI input 0 1 An overrun has occurred on NMI input 0
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 NIF1	NMI Status Flag 1 (main core) If enabled (i.e., NCR[NREE1] or NCR[NFEE1] is set), an NMI input event will set NIF1. NIF1 causes an interrupt request based on the NMI destination, as configured by NCR[NDSS1], and/or a system wakeup request based on NCR[NWRE1]. 0 No event has occurred on the pad 1 An event as defined by NCR[NREE1] and NCR[NFEE1] has occurred
9 NOVF1	NMI Overrun Status Flag 1 (main core) This field is a copy of the current NIF1 value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. NOVF1 causes an interrupt request based on the NMI destination, as configured by the NCR[NDSS1] bits, and/or a system wakeup request based on the NCR[NWRE1] bit configuration. 0 No overrun has occurred on NMI input 1 1 An overrun has occurred on NMI input 1
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 NIF2	NMI Status Flag 2 (IOP)

Table continues on the next page...

WKPU_NSR field descriptions (continued)

Field	Description
	This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE or NFEE set), an NMI input event will set NIF. NIF causes an interrupt request based on the NMI destination, as configured by the NDSS bit, and/or a system wakeup request based on the NWRE bit configuration. 0 No event has occurred on the pad 1 An event as defined by NCR[NREE2] and NCR[NFEE2] has occurred
17 NOVF2	NMI Overrun Status Flag 2 (IOP) This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (RREE or RFEE set), ROVF can cause a reset request to MC_RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short "functional" reset. 0 No overrun has occurred on NMI input 2 1 An overrun has occurred on NMI input 2
18–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RIF	Reset Status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (i.e., if RREE or RFEE is set), RIF status bit is set and will cause a functional reset request to RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short "functional" reset. 0 No event has occurred on the pad 1 An event as defined by NCR[RREE] and NCR[RFEE] has occurred
25 ROVF	Reset Overrun Status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (RREE or RFEE set), ROVF can cause a reset request to MC_RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short "functional" reset. 0 No overrun has occurred on the NMI reset request input 1 An overrun has occurred on the NMI reset request input
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

90.3.2 NMI Configuration Register (WKPU_NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

NOTE

Writing a '0' to both NREE*n* and NFEE*n* disables the NMI functionality completely (i.e. no system wakeup or interrupt will be generated on any pad activity).

NOTE

Writing to the NCR when NIF = 1 or NOVF = 1 causes an interrupt to the destination source/system wakeup to be generated independent of an NMI wakeup event.

Address: FEED_0000h base + 8h offset = FEED_0008h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NLOCK0	NDSS0		NWRE0	0	NREE0	NFEE0	NFE0	NLOCK1	NDSS1		NWRE1	0	NREE1	NFEE1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NLOCK2	NDSS2		NWRE2	0	NREE2	NFEE2	0	RLOCK	RDSS		RWRE	0	RREE	RFEE	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

WKPU_NCR field descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0 0 Writing 0 to this bit has no effect. 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset.
1–2 NDSS0	NMI Destination Source Select 0 00 A non-maskable interrupt will occur if NSR[NIF0] or NSR[NOVF0] is set. 01 A critical interrupt will occur if NSR[NIF0] or NSR[NOVF0] is set. 10 A machine check request will occur if NSR[NIF0] or NSR[NOVF0] is set. 11 No reaction
3 NWRE0	NMI Wakeup Request Enable 0 0 System wakeup requests from the corresponding NSR[NIF0] bit are disabled 1 A set NSR[NIF0] bit or set NSR[NOVF0] bit causes a system wakeup request
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 NREE0	NMI Rising-edge Events Enable 0 0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 NFEE0	NMI Falling-edge Events Enable 0 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 NFE0	NMI Filter Enable Enable analog glitch filter on the NMI pad input.

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
	0 Filter is disabled 1 Filter is enabled
8 NLOCK1	NMI Configuration Lock Register 1 0 Writing 0 to this bit has no effect. 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset.
9–10 NDSS1	NMI Destination Source Select 1 00 A non-maskable interrupt will occur if NSR[NIF1] or NSR[NOVF1] is set. 01 A critical interrupt will occur if NSR[NIF1] or NSR[NOVF1] is set. 10 A machine check request will occur if NSR[NIF1] or NSR[NOVF1] is set. 11 No reaction
11 NWRE1	NMI Wakeup Request Enable 1 0 System wakeup requests from the corresponding NSR[NIF1] bit are disabled 1 A set NSR[NIF1] bit or set NSR[NOVF1] bit causes a system wakeup request
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 NREE1	NMI Rising-edge Events Enable 1 0 Rising-edge event is disabled 1 Rising-edge event is enabled
14 NFEE1	NMI Falling-edge Events Enable 1 0 Falling-edge event is disabled 1 Falling-edge event is enabled
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 NLOCK2	NMI Configuration Lock Register 2 0 Writing 0 to this bit has no effect 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset.
17–18 NDSS2	NMI Destination Source Select 2 00 A non-maskable interrupt will occur if NSR[NIF2] or NSR[NOVF2] is set. 01 A critical interrupt will occur if NSR[NIF2] or NSR[NOVF2] is set. 10 A machine check request will occur if NSR[NIF2] or NSR[NOVF2] is set. 11 No reaction
19 NWRE2	NMI Wakeup Request Enable 2 0 System wakeup requests from the corresponding NSR[NIF2] bit are disabled 1 A set NSR[NIF2] bit or set NSR[NOVF2] bit causes a system wakeup request
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 NREE2	NMI Rising-edge Events Enable 2 0 Rising-edge event is disabled 1 Rising-edge event is enabled

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
22 NFEE2	NMI Falling-edge Events Enable 2 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RLOCK	Reset Configuration Lock Register 0 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . 1 Writing a 0 has no effect.
25–26 RDSS	Reset Destination Source Select 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
27 RWRE	Reset Wakeup Request Enable 0 System wakeup requests from NSR[RIF] are disabled 1 A set NSR[RIF] bit or set NSR[ROVF] bit causes a system wakeup request
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 RREE	Reset Rising-edge Events Enable 0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 RFEE	Reset Falling-edge Events Enable 0 Falling-edge event is disabled 1 Falling-edge event is enabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

90.4 Functional Description

This section provides a functional description of the WKPU features.

90.4.1 Non-Maskable Interrupts

The WKPU allows the external NMI pin (ESR1) to assert the non-maskable interrupts of the cores on the device. These interrupts include non-maskable interrupts, critical interrupts, or machine check requests to any of the cores including the main core, IOP,

and safety core. The safety core's non-maskable interrupt line from the WKPU is combined with the FCCU internal NMI interrupt event arising due to fault management logic.

The NMI destination interrupt is controlled by the user through the configuration of NCR[NDSS*n*].

The enabling of rising-edge, falling-edge, or either-edge reaction to the NMI pin is independent for each core, and is controlled by NCR[NFEE*n*] and NCR[NREE*n*].

The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

The NMI supports a status flag, NSR[NIF*n*], and an overrun flag, NSR[NOVF*n*].

90.4.2 Reset management

The WKPU allows the external NMI pin (ESR1) to assert a non-destructive reset request to the MC_RGM module. The ESR1 reset request can be reconfigured in the MC_RGM to generate a SAFE mode or interrupt request (see "Reset Generation Module (MC_RGM)" for details).

The enabling of rising-edge, falling-edge, or either-edge reaction to the NMI pin is independent for the reset request, and is controlled by the RFEE and RREE bits in the NCR register.

Reset request supports a status flag and an overrun flag which are located in the NSR register. See NMI Configuration Register (WKPU_NCR) for more information.

90.4.3 System Wakeup

The WKPU allows the external NMI pin (ESR1) to assert a system wakeup from STOP to the MC_ME module on the chip. The system wakeup is enabled through the NWRE or RWRE bits in the NCR register.

90.4.4 Glitch Filter

An analog glitch filter, between the NMI pin (ESR1) and the WKPU module, is enabled by NCR[NFE0]. See the chip's data sheet for the glitch filter's electrical specifications.

90.4.4.1 Edge Detect

Each NMI and reset request can be enabled or disabled independently. This can be performed using the NCR register, which contains all configuration bits for a given NMI in a single byte. The user can configure interrupts with an active rising edge, an active falling edge, or either edge being active. Disabling both edge events results in no interrupt or reset being propagated from the WKPU block.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits. The active reset request edge is controlled by the RREE and RFEE bits.

NOTE

After reset, all rising-/falling-edge enable bits set to 0, therefore the NMI functionality and reset request is disabled after reset and must be enabled explicitly by software.

90.5 Initialization Information

This section discusses initialization for the following features:

- Glitch Filter and Pad Configuration
- Non-maskable interrupt initialization
- Reset Request

90.5.1 Glitch Filter and Pad Configuration

Glitch filter control and pad configuration should be performed while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

When enabling the glitch filter, do not enable the rising-/falling-edge events bits, i.e., the NREE, NFEE, RREE, and RFEE, bits in the same register write.

90.5.2 Non-maskable interrupt initialization

When an NMI interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization the user must do the following:

1. Mask interrupts by clearing NSR[NIF*n*] and NSR[NOVF*n*].
2. Clear NCR[NFEE*n*] and NCR[NREE*n*], preferably by writing 0 to all bits in the register.
3. Configure the analog glitch filter using NCR[NFE0].
4. Configure the appropriate SIUL2_MSCR register for the desired NMI interrupt pin(s) as follows:
   - Clear the ODC bits to disable output.
   - Set the IBE bit to enable the pin's input buffer.
   - If the internal weak pull-up/pull-down is used, configure the appropriate WPUE and WPDE bits.

NOTE

NMI interrupt pins should never be configured as outputs, i.e., MSCR[ODC] bits are not zeros, when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

5. Write appropriate bit values to the NCR register to configure:
   - NMI source (NDSS*n*).
   - Wakeup request (NWRE*n*).
   - Glitch filter (NFE0)
   - Rising/falling edge events enable (NFEE*n* and NREE*n*).
   - NMI configuration lock (NLOCK*n*).

90.5.3 Reset Request

When an NMI reset request pin is first enabled, it is possible to get a false reset request. To prevent a false reset request during pin reset request initialization, the user must do the following:

1. Mask reset requests by clearing NSR[RIF] and NSR[ROVF].

2. Clear NCR[RFEE] and NCR[RREE], preferably by writing 0 to all bits in this register.
3. Configure the analog glitch filter, as desired, using NCR[NFE0].
4. Configure the appropriate SIUL2_MSCR register for the NMI interrupt pin(s) as follows:
   - Clear the ODC bits to disable output.
   - Set the IBE bit to enable the pin's input buffer.
   - If the internal weak pull-up/pull-down is used, configure the appropriate WPUE and WPDE bits.

NOTE

NMI reset request pins should never be configured as outputs (i.e., MSCR[ODC] bits are not zeros) when external reset request inputs are desired since false reset requests could be detected (e.g., from a GPIO configuration).

5. Write appropriate bit values to the NCR register to configure:
   - Reset destination source (RDSS]).
   - Reset wakeup request (RWRE).
   - Glitch filter (NFE0)
   - Rising/falling edge events enable (RFEE and RREE).
   - Reset configuration lock (RLOCK).

Appendix A
Release Notes for Revision 4

A.1 General changes

• No substantial content changes

• In the second "Core Debug Support" chapter, added the name of the core (e200z720n3) throughout the chapter.
• In the "IRCOSC Digital Interface" chapter, in the "Functional description" section, deleted the second paragraph and bulleted list pertaining to the power-on reset sequence.
• In the PSI5 chapter:
  • In PSI5S_LINCR2, footnote “Register bit can be read/set by software” changed to a NOTE in the ABRQ field description.
  • In PSI5S_PTD register description, corrected missing cross-reference in the last sentence.
  • In PSI5S_GLSR register, modified bit position of DIRCMD_RDY field.
  • Removed PSI5S_WD_DDTRIG_CLKDIV0, PSI5S_WD_DDTRIG_CLKDIV1, PSI5S_WD_DDTRIG_CLKDIV2, and PSI5S_WD_DDTRIG_CLKDIV3 registers.
  • Added PSI5S_DIRCMD register.
  • In PSI5S_E2SCR_CHn, redundant footnote on start condition removed. (The description is present in the CMD_TYPE field description.)
• Editorial (non-technical) changes and improvements throughout.
• Spelling correction and wording improvements.

A.2 Preface changes

• No substantial content changes

A.3 Introduction changes

In section, Block diagrams, • Updated Periphery allocation figure with two SIPI module. • Updated Block diagram figure with one more LFAST and SIPI block. • Updated Flash size to 8640 KB in sections, Features summary and Feature list.
• In the Software debug and calibration section, changed overlay RAM to 1M byte.

A.4 Embedded Memories changes

• Editorial changes.
• Chip-specific C55FMC flash register information :   • New section contains device-specific C55FMC_LOCKn and C55FMC_SELn register definitions.
• In Flash memory array, changed the flash size (was 8 MB, is 8640 KB).

A.5 Signal Description changes

• No substantial content changes

A.6 Memory Map changes

Editorial changes.

A.7 Functional Safety changes

• In Clock modified cross-reference to MCU clocking details
• In Table 1 deleted column “Recommended recovery: Destructive reset"
• Minor editorial change
• In Online Logical BIST (LBIST), updated the note to, "When Scan mode is entered through LBIST, the flash memory must be in IDLE state, therefore Program or Erase operations must not be either active or suspended."
• Removed section "FCCU failure inputs". This section is now located in the FCCU Device Configuration section of the RM.
• Editorial updates.
• Updated contents of Table 6-2.
• Deleted requirement tag "[Covers: Saf1107]" from topic End-to-End protection on data path. • Changed TT_CAN to M_TTCAN throughout the module. • In Table 6-3 changed DMACHMUX to DMAMUX. • Updated topic Dual-core lockstep • Updated Table 6-5. • In Interface to ECC units, updated paragraogh that starts with "Not all ECC units provide...".

A.8 Device Configuration Changes

• Added Error signal flow diagram.
• In the FCCU bridge topics, FCCU register availability and reset values table, added row for FCCU_MCS register with a reset value of 0x0000_8083.
• In the FCCU Device Configuration, added section "FCCU failure inputs" (FCCU failure inputs).
• In Decorated Storage Memory Controller (DSMC), added footnote to DSMC instances associated with processor cores: "Instances of the DSMC associated with processor core tightly-coupled memory do not protect both ports—only the backdoor port used by other cores or peripherals is protected. Any IMEM or DMEM location expected to be referenced with atomic access should only be accessed by the local processor core via a decorated storage instruction."
• In Interrupt sources, removed Interrupt Sources, 490 and 491. Updated MC_ME3 interrupt source to include ME_IS[I_ICONF] | ME_IS[I_ICONF_CC] | ME_IS[I_ICONF_CU]. • In Table 7-67, updated 'Nominal value expected' for LVD270_C, LVD270_F, LVD295_F, LVD295_A, and LVD400_A.
• In Table 7-23, added extra sources to IRQs 558, 559, 561, and 562.
Changed the following in the DSPI Instantiation table: • Changed DSPI2 value in LVDS column from "Not Available" to "Available." • Added column for CMD FIFO Depth.
Changed the FCCU Channel 5, SSCM_XFER_FLASH_ERR, the "Error reaction path check" cell from "Not testable" to "Testable." Replaced the following in FCCU Channel 53: VDD_HV_IO_JTAG, VDD_HV_IO_FLEX and VDD_HV_IO_MAIN" with "VDD_HV_IO_EBI, VDD_HV_IO_FLEX, and VDD_HV_IO_FLEXE."

• FCCU failure inputs
  • In Table 7-86
  • Changed 'FCCU channel' 27 to Reserved.
  • Changed 'FCCU channel' 28 to Reserved.

• PMC ADC test mode
  • In Table 7-67,
  • Row 6b101101, updated 1.28V to 1.30V in Nominal value field.
  • Row Internal POR on flash HV, updated LVD_Threshold to POR250_Threshold in Nominal value field.
  • Row LVD400_IM, updated LVD_Threshold to LVD400_Threshold in Nominal value field.
  • Rows LVD108_F/P, added footnote for Nominal value expected
  • Added footnote, "bandgap trimmed reference voltage" for all of equations with '~supply X 1.2/*_Threshold'.

• Added CRS register reset values for XBAR_0 and XBAR_1 configurations.
• In PMC ADC test mode updated number of channels to 110 that are used for PMC analog signal monitoring.
• Added Wait time for writing to the online registers.

A.9 Reset and Boot changes

• No changes.

A.10 "Device Configuration Format (DCF) Records

- Revised 4th step in the sequence describing behavior after power is applied in Introduction.
- Added DCF records and UTEST DCF records.
- Miscellaneous DCF registers: Changed title from "DCF registers" to "Miscellaneous DCF registers." Added JTAG Pins table/description. Updated JTAG Pin Configuration DCF Client offset to 000000000000111.
- Reversed order of bit fields in "PMC_REE bits DCF Client" table.
- DCF client table: Content revisions. Added description for DCF order dependency in flash memory.
- DCF records: Changed OTP UTEST flash memory start address to 0x0040_0300.

- Added xref to XOSC Start-up in the Table 9-8 XOSC_LOAD_CAP_ SEL field description.

- Made editorial and technical-content changes to Introduction.
- Made substantial changes to:
  - Table 9-6
  - Miscellaneous DCF registers
- Table 9-8 :
  - Added a note to XOSC_LF_EN description.
  - Changed OSC_EN_40MHZ field description.
  - Changed ESR0_GATE field description.
- Table 9-7 : deleted "Reserved" from XOSC_IBIAS_SEL field.

- Updated "PMC_REE bits DCF Client" table with HVD8_F bit field.

A.11 Power Management changes

- Editorial changes.
- In Checker core disabling, added "(unless user intentionally disables checker clock, e.g. afterrun mode)."
- In Flash power requirements, added content to the last paragraph.
- Revised Table 10-4.
- In VDD_LV conditions and LVD trimming/enabling sequence, changed "After analog delay (tLVDTRIM) has elapsed" to "After a delay (including tLVDTRIM and other system operations) has elapsed".
- In Power-down sequence, added "In the case of an intentional power down of the Low Voltage supply (1.2V), the Auxillary Regulator should be disabled by writing a 1 to the bit VREG1P2_CTRL[VREG1P2_DIS]."
- Updated Figure 10-4.
- Updated Figure 10-6.
- Updated Figure 10-7.
- Updated Figure 10-8.
- Revised VDD_HV conditions and LVD trimming/enabling sequence.

A.12 Security changes

- In Basic security, replaced BAM with BAF.

A.13 Calibration_and_Debug changes

Changed e200z720n3 core name to e200z710n3.
• No substantial content changes.
• Replaced 'DCI cross triggering' entry under the 'Signal' column, to 'Debug Break and Cross Triggering'.
• Added JTAGC ACCESS_AUX block instructions.

A.14 Core_Complex_Overview changes

• No substantial content changes

A.15 Core description module changes

• In Figure 14-1, ensured that the heading "Debug Registers" appears without a subsequent footnote marker.
• Revised Figure 14-1 and Figure 14-2.

• Hardware Implementation Dependent Register 0 (HID0) :
  • New
• Hardware Implementation Dependent Register 1 (HID1) :
  • New

• Register model :
  • Correction to Supervisor Mode Programmer's Model SPRs figure: Added Thread ID register (TIR) and System Information Register (SIR)
  • Correction to Supervisor Mode Programmer's Model DCRs and PMRs figure: Added Thread Management Register TMCFG0
  • Correction to User Mode Programmer's Model SPRs figure: Added System Information Register (SIR)
• Instruction timing :
  • New
• Exceptions :
  • Footnote added to Critical input offset value in Exceptions and conditions table
• Machine Check Syndrome Register (MCSR) :
  • Footnote added to Exception type column heading in field descriptions table
  • Footnote added to DC_DPERR in field descriptions table

A.16 Core description module changes

• Register model :
  • Figure 15-1 : Added Thread ID register (TIR) and System Information Register (SIR)
  • Figure 15-1 : L1CSR2 register deleted
  • Figure 15-2 : Added Thread Management Registers (TMR)
  • Figure 15-2 : Deleted Cache Access Registers

• Figure 15-3 : Added System Information Register (SIR) • Figure 15-3 : Added footnote to Nexus PID • Added "Instruction timing" section.
• DMEM Control Register 0 (DMEMCTL0) : • For DMEMCTL0 register, corrected DSWCE=11b description for local memory partial-width slave write checking • IMEM Control Register 0 (IMEMCTL0) : • For DMEMCTL0 register, corrected DSWCE=11b description for local memory partial-width slave write checking • End-to-End ECC fault injection by software : • In E2EECSR0[RCHKBIT] description, added information about external read accesses that receive an error response.
• End-to-End ECC fault injection by software : • Changed "error generation" to "fault injection" • Added description for external read accesses that receive an error response
• Added "Cache" section and subsections.
• In Table 15-18 of Machine Check Syndrome Register (MCSR) • In table footnote about MEA, corrected core name • Added footnote for exception type
• Machine State Register (MSR) : • Bits added to register figure and field descriptions table: • SPV (SP/Embedded FP/Vector available) • WE (Wait State (Power management) enable) • FP (Floating-Point Available) • FE0 (Floating-point exception mode 0) • FE1 (Floating-point exception mode 1) • IS (Instruction Address Space) • DS (Data Address Space) **NOTE:** None of the above bits are functional. They were previously shown as reserved bits but are redefined to clearly show any deviations from Book III-E of the Power ISA version 2.06 specification.
• Register model : • Correction to Supervisor Mode Programmer's Model SPRs figure: "LSP" removed from EFPU Registers section • Correction to Supervisor Mode Programmer's Model DCRs and PMRs figure: Thread Management Register (TMR16) name changed to TMCFG0 (was DMEMCTL0)
• Hardware Implementation Dependent Register 0 (HID0) : • New • Hardware Implementation Dependent Register 1 (HID1) : • New
• DMEM Control Register 0 (DMEMCTL0) : • First sentence of DSWCE field value 11 description changed to, "Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte) writes." (was "...(1-, 2-, or 3-byte) writes.")

A.17 SIUL2 module changes

• Updated number of external interrupts (EIRQs) to 10, 341 pads and 22 ports throughout. • Updated SIUL2 block diagrams. • In Section, External interrupt request: Added EIRQ 3 to 'interrupt request 0'.

A.18 XBAR changes

- In XBAR Priority Registers Slave (XBAR_PRS*n*), added the note "See Chip Configuration section for this device's reset values." at the beginning of the section.
- In XBAR Control Register (XBAR_CRS*n*), changed the reset value (was 0000_0000h, is 00FF_0000).

A.19 XBIC module changes

- No substantial content changes

A.20 Peripheral Bridge changes

- General operation :
  - Following note added: “FlexRay access to peripherals via the peripheral bridge is not supported“.

A.21 SMPU changes

- No changes.

A.22 IAHBG changes

- In Timing modes, removed "32 -> 64 interface" and "64 -> 32 interface" sections.

A.23 SEMA42 module changes

- Added AUTOSAR definition to Introduction.

- Corrected register prefix to SEMA42 throughout chapter.
- Made editorial and/or technical changes to:
  - Gate Register (SEMA42_GATE*n*) register description.
  - Reset Gate Write (SEMA42_RSTGT_W) register description.
  - RSTGDP field description.
  - Reset Gate Read (SEMA42_RSTGT_R) replaced register description with a sentence referring to SEMA42_RSTGT_W description.

• RSTGSM description for value 10. • Functional description : minor edit. • Multi-core programming 101: software gates : Changed "writes" to "accesses" in the first paragraph. While one processor updates/writes data, another one must not read it.
• Edited two citations in Multi-core programming 101: software gates.

A.24 INTC module changes

• Completely replaced Examining LIFO contents. • In Raised priority preserved, replaced the timing diagram and associated table of event descriptions.
• In Raised priority preserved, deleted the text "Also, during the epilog of the interrupt exception handler for this preempting ISR, the raised priority has been restored from the LIFO to PRI in INTC_CPRn."
• In Interrupt sources, changed "chapter that describes how modules are configured and connected" to "chip-specific INTC information".
• In INTC Master Protection Register (INTC_MPROT), in the ID field description, added "Only the field values related to core bus masters can be selected. Non-core bus masters cannot be selected and would result in a bus access error."
• In Block diagram, revised the stem sentence and figure title to indicate that the diagram is an example of a four-processor configuration, and that the actual number of supported processors is given in the chip-specific INTC information. • In Features, ensured that the bulleted list shows only those processors supported on this chip. • Revised Memory map and register definition to show only those registers and fields actually present on this chip.
• In Memory map and register definition, changed "Although INTC_SSIn are 8 bits wide..." to "Although INTC_SSCIRn are 8 bits wide...".
• In INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*), changed "... can write to the CLR bit and not the SET bit, otherwise ..." to "... can write to the CLR bit, otherwise ...".

• In Memory map and register definition :
  • Added Figure 23-2 and surrounding text explaining write restrictions for PSR*n* and SSCIR*n*.
  • In INTC Priority Select Register (INTC_PSR*n*) and INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*), added "See Figure 23-2 for limitations on writing to this register."

A.25 eDMA changes

• In Enable Error Interrupt Register High (eDMA_EEIH) and Interrupt Request Register High (eDMA_INTH), deleted the extraneous text "[[MAT+MCK]]".
• In TCD Control and Status (eDMA_TCD*n*_CSR), changed “XBS” to “XBAR“ in BWC field description. • In Control Register (eDMA_CR), the reset value was changed from 0x0000_0000 to 0x0000_E400. • In Control Register (eDMA_CR), bit 31 was changed from read-only to read-write.

A.26 DMAMUX module changes

• Updated note in the register description for Channel Configuration register (DMAMUX_CHCFGn) as follows: "Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0)."

- In the section "Enabling and configuring sources," changed code examples, from #define DMAMUX_BASE_ADDR 0xFC084000/* Example only ! */ to #define DMAMUX_BASE_ADDR 0x40021000/* Example only ! */.

A.27 Clocking changes

- Updates to Figure 26-1 in Clock generation :
  - Added PSI5-S (PSI5_S_BAUD_CLK) to AUX Clock 0 Divider 4
  - Added footnote for PSI5_S_BAUD_CLK: "Referred to as ipg_baud_clk in the PSI5-S chapter."
  - Minor non-technical changes
- Update to Table 26-1 in MC_CGM registers :
  - New output clock added to CGM_AC0_DC4, CGM_AC0_SC regusters: PSI5-S (PSI5_S_BAUD_CLK)
- Update to Table 26-2 in System clock frequency limitations :
  - Added PSI5_S_BAUD_CLK (PSI5-S) to divider logic list for Aux Clock 0 Selector (Divider 4)

- Updated XOSC_CTL register reset value to 0030_8000h
- Updated IRCOSC_CTL register reset value to 0000_0000h

- Updated Divider1(DIV1) value from '1...1024' to '1...256' in figure, Figure 26-2

- Added explanation of PSI5-S clock in PSI5 clock dividers.

- Loss of IRCOSC clock
  - Updated Figure 26-11, adding the FCCU and FOSU modules and their connections.

A.28 PLLDIG changes

- DTHDIS
  - Added note (Important) "This field must be written 00b" to the DTHDIS field description.

- Features
  - Removed bullets showing PLL input/output frequencies. Duplicate information that is the Data Sheet.

- PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)
  - Added the requirement ID PLLDIG_BLG_005 (enclosed in square brackets) to the beginning of the LOLIE field description.
  - Added "end" in square brackets to the end of the LOLIE field description.
  - Updated reset value of reserved field 21, and changed access to RO.
- PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)
  - Added the requirement ID PLLDIG_BLG_006 (enclosed in square brackets) to the beginning of the LOLF field description.
  - Added "end" in square brackets to the end of the LOLF field description.
- PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)
  - Added the requirement ID PLLDIG_BLG_007 (enclosed in square brackets) to the beginning of the LOLIE field description.
  - Added "end" in square brackets to the end of the LOLIE field description.
- PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)
  - Added the requirement ID PLLDIG_BLG_008 (enclosed in square brackets) to the beginning of the LOLF field description.
  - Added "end" in square brackets to the end of the LOLF field description.

- Added new topic Maximum lock time.

- Clock configuration
  - Updated equation for f_{PLL0_VCO} by adding a x2 multiplier in numerator.

- Editorial updates throughout.
- Clock configuration

• Updated equations in section.
• PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)   • Updated 'PLL0' to 'PLL1' in Register description, "...effective after PLL1 is disabled, then reenabled."

• PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)
  • Replaced the sentence, "The values of PREDIV and MFD..." with "PLL0DV can be modified at anytime..." in the register description.
• PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)
  • Updated register description.
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)
  • Removed the sentence, "Changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] fields..." in the register description.
• Clock configuration
  • Add equation headings and titles to equations.
• Frequency modulation
  • Added "Equation" titles to formulas.
  • Updated Figure 27-1 to show that MD is a peak-to-peak value.

• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)
  • Removed the text ", however, the output clock from PLL..." from the last sentence in the register description.
• Normal mode with reference, PLL0 or both PLLs enabled
  • Replaced "output divider (Reduced Frequency Divider) and whether PLL modulating is enabled" with "and modulation enable are" in second paragraph.
  • Updated the end of the first sentence, "In normal mode, PLL0 receives..."
• Introduction
  • Removed "The chip provides a user interface and control over" from the first sentence which now reads "The Dual PLL system composed of..."

• PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)
  • Added updated content to the register description.
• PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)
  • Added updated content to the register description.
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)
  • Removed the text "the binary equivalent of" from the field descriptions of INCSTP and MODPRD.

• Removed the section "Acronyms and Abbreviations". • Frequency modulation   • Updated the equation in the Note, from "$f_{max} = f_{sys} \times (1 + (MD\%)$." to "$f_{max} = f_{sys} \times (1 + (MD\% / 100))$."
• Frequency modulation   • Updated the sentence "If center modulation is selected..." to "If center modulation is used..." in the Note "The device maximum operating frequency..."
• Register PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)   • Replaced the register description with updated content.
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)   • Updated MODSEL bitfield value descriptions for both 0 and 1.

A.29 CMU changes

• CMU Interrupt Status Register (CMU_ISR)   • Added NOTE to the OLRI field description, "While entering STOP mode, OLRI may be triggered. To avoid this when the system is running on the PLL, disable the XOSC before entering STOP mode."

• Introduction
  • Added NOTE "See the 'Clocking' chapter for chip-specific sources used by the CMU."
• CLKMN1 supervisor
  • Added the NOTE "An example of determining the..."

• CLKMN1 supervisor

- Updated NOTE to read "($HFREF_{Actual}$ = ($HFREF_{Ideal}$ ÷ 0.95)" instead of "($HFREF_{Actual}$ = ($HFREF_{Ideal}$ + 2) × 1.05)".
- Added "The actual LFREF value will be 762 when the accuracy is taken into consideration ($LFREF_{Actual}$ = ($LFREF_{Ideal}$ ÷ 1.05)." to the NOTE.

A.30 MC_CGM module changes

- In Memory map and register definition,
  - added MC_CGM_AC5_CDC23 register.
  - Updated reset values of registers, MC_CGM_AC5_CDC0, MC_CGM_AC5_CDC1, MC_CGM_AC5_CDC2, MC_CGM_AC0_SS, and MC_CGM_AC0_SC.

- Editorial changes.
- Added note to the DIV_FMT value in the MC_CGM_AC0_DC3 register to select the correct DIV_FMT value.

- Updated MC_CGM_AC10_SC[SELCTL] field description.
- In Memory Map and Register Definition, Updated note, 'All registers will be accessible in "User mode"... only if SSCM_ERROR[RAE] is set'.

A.31 XOSC changes

- Functional description
  - Added "(from MC_ME)" in the first column of heading row of Table 30-1.

- Added note to clarify bypass operation.

- XOSC Control Register (XOSC_CTL)
  - Split the Reserved bitfield at offset 7 into two Reserved fields of 5 bits and 2 bits.

A.32 IRCOSC changes

- No substantial content changes

A.33 PRAMC module changes

- Reliability considerations
  - Changed "Safety considerations" in section title and content to "Reliability considerations"
- Transaction monitor
  - Changed the sentence, "Functional safety coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies...." to "Fault detection coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies...."

- SRAM controller memory map and register definition:
  - Changed the first sentence from, "The RAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot" to "...programming model mapped to a standard on-platform peripheral slot".

• SRAM controller memory map and register definition :
  • Note added regarding caution required when reconfiguring platform RAM controller during device operation
• Hsiao ECC algorithm :
  • Reformatted RAM controller basic H-matrix definition table. Added new column, XOR Sum, as placeholder for calculated checkbit. The XOR sum checkbit for each row is calculated by XORing all values in the positions marked with an asterisk in that row.

• Less than 64-bit writes :
  • Added single bit and multi bit error signals to Read-modify-write data path figure
  • Changed hardware signal names to names that describe signal function more clearly

• Optional read wait-state :   • All references to the PRCR1 register changed to "PRCRx". This is only an editorial change--there is no change to any register name or function.

A.34 Flash memory controller module changes

• Reliability considerations :
  • Section renamed to "Reliability considerations" (was "Safety considerations")
  • Text added: "This section summarizes mechanism to enhance the reliability of Flash by correcting and detecting faults."

• Flash address generation check :   • Clarification: "Functional safety" changed to "Fault detection"
• Overlay RAM feedback check :   • Clarification: "Functional safety" changed to "Fault detection"
• Reliability considerations on overlay accesses :   • Section renamed to "Reliability considerations on overlay accesses" (was "Safety considerations on overlay accesses")
• PFlash calibration remap to RAM :   • Editorial fixes (no technical changes)
• Censorship :   • "Security flash blocks" added to list of flash regions

• Error termination :
  • "...returns an error response on an attempted access ", changed to, "returns an error response on an attempted access to a partition that is unavailable."
  • Link to Embedded Flash Memory chapter added

• Platform Flash Remap Control Register (PFLASH_PFCRCR) :   • Caution note added explaining that the use of overlay remap functions has a potential for causing data coherency issues *when used with line read buffers enabled*. To prevent issue, clear line read buffers prior to enabling any remap region.
• Functional description :   • Deleted from first sentence: "As shown in the block diagram"

• ECC on data flash accesses :
  • Editorial changes
  • Clarification: Following note added: "EEPROM should be avoided for storage of executable code."

• Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRDn_Word0) :
  • Clarification: Previously the LSTARTADDR field was shown as a 32-bit field with read/write access. The address is a 32-bit address but the least significant 4 bits are always 0 and are now shown as a reserved read only zero field.
• Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRDn_Word1) :
  • Clarification: Previously the PSTARTADDR field was shown as a 32-bit field with read/write access. The address is a 32-bit address but the least significant 4 bits are always 0 and are now shown as a reserved read only zero field.

• Platform Flash Configuration Register 1 (PFLASH_PFCR1) :   • Clarification: All "1xx" values in the APC field have the same effect. Previously, the values, 100, 101, 110, and 111 in the APC field were listed individually and had identical descriptions. They are now condensed into a single entry, "1xx" in the value list to reduce the possibility of confusion.
• Platform Flash Configuration Register 1 (PFLASH_PFCR1) :   • Note added to PFLASH_PFCR1[APC] field description: "A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup."

A.35 c55fmc module changes

• Added note about reset value to Alternate Module Configuration Register (C55FMC_MCRA) description: "Reset values are dependent on chip configuration. See the Chip Configuration chapter for specifics."
• Clarified information about chip-specific reset value of:
  • Extended Module Configuration Register (C55FMC_MCRE)
  • Over-Program Protection 0 register (C55FMC_OPP0)
  • Over-Program Protection 1 register (C55FMC_OPP1)
  • Over-Program Protection 2 register (C55FMC_OPP2)
  • Over-Program Protection 3 register (C55FMC_OPP3)

Features • Clarification: Feature "Triple Voted Flops for safety critical flash functions" changed to "Triple Voted Flops for flash functions requiring high reliability".
Test mode disable seal • Detail added: "UTest operations Margin Read and Array Integrity are always protected even if the Test Mode Disable Seal password is written."
Features • Clarification: Feature "Triple Voted Flops for flash functions requiring high reliability" changed to, "Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control"

A.36 DSMC module changes

• No substantial content changes

A.37 Flash Memory Programming and Configuration changes

• Removed incorrect content from the "Creating password groups" section.

A.38 EBI module changes

- In Modes of Operation, replaced EBI_BR3 with EBI_BR2.
- Memory Map and Register Definition,
  - Module Configuration Register (EBI_MCR): Updated reset value to 0000_0800h. Updated bit field access EBI_MCR[20:19] to RW.
  - Updated the numbers of instances for EBI_BR[0:2] and EBI_OR[0:2] registers.

A.39 ADC Configuration changes

- In Table 38-21, changed input channel 119 from a writeable field to Reserved in the following registers:
  - TCIPR
  - TCIMR
  - TCDSR
  - TCNCMR
  - TCJCMR
  - TCWSELR2
  - TCWENR
  - TCAWORR
- In Table 38-22, renamed 'SARADCB_ECIPR0' register to 'SARADC_ECDSDR'.

- Simultaneous start of multiple Sigma-Delta converters : Updated diagram Figure 38-4.

- Internal voltage monitor channel assignments
  - Added footnote "When doing conversions on this channel, it is recommended to mask the corresponding HVD/LVD from generating a reset for the duration of the conversion when this channel is enabled." to the column "SARADC_B input channel" and rows with channels 112...118, Table 38-16.

- Internal voltage monitor channel assignments
  - Updated the row of channel 111 row making this channel Reserved, in Table 38-16.

- In Table 38-1, removed unused pins from the entry for the SARADC_B instance.

- Expanded Table 38-20 to show channels 64..87 for ICWSELR, ICAWORR, and ICWENR.

- SARADC_0 register definitions
  - In table, updated the row of SARADC0_WTHRHLR0-WTHRHLR3 to SARADC0_WTHRHLR[0:3].
  - In row SARADC0_ICWSELR0, changed width of fields WSEL_CH7, WSEL_CH6, WSEL_CH5, WSEL_CH4 from 3 bits to 2 bits.
- SARADC_1 register definitions
  - In table, updated the row of SARADC1_WTHRHLR0-WTHRHLR3 to SARADC1_WTHRHLR[0:3].
  - In row SARADC1_ICWSELR1, changed width of fields WSEL_CH15, WSEL_CH14, WSEL_CH13, WSEL_CH12, WSEL_CH11, WSEL_CH10, WSEL_CH9, WSEL_CH8 from 3 bits to 2 bits.
- SARADC_2 register definitions
  - In table, updated the row of SARADC2_WTHRHLR0-WTHRHLR3 to SARADC2_WTHRHLR[0:3].
  - In row SARADC2_ICWSELR3, changed width of fields WSEL_CH23, WSEL_CH22, WSEL_CH19, WSEL_CH18, WSEL_CH17, WSEL_CH16 from 3 bits to 2 bits.
- SARADC_3 register definitions
  - In table, updated the row of SARADC3_WTHRHLR0-WTHRHLR3 to SARADC3_WTHRHLR[0:3].
  - In row SARADC3_ICWSELR3, changed width of fields WSEL_CH31, WSEL_CH30, WSEL_CH29, WSEL_CH28, WSEL_CH27, WSEL_CH26 from 3 bits to 2 bits.
- SARADC_4 register definitions
  - In table, updated the row of SARADC4_WTHRHLR0-WTHRHLR3 to SARADC4_WTHRHLR[0:3].
  - In row SARADC4_ICWSELR4, changed width of fields WSEL_CH32, WSEL_CH33, WSEL_CH34, WSEL_CH35 from 3 bits to 2 bits.

A.40 SDADC module changes

- Module Configuration Register (SDADC_MCR)
  - Notes added to SDADC_MCR[PGAN] field description and SDADC_CDR register description noting that when programmable gain is 16 (MCR[PGAN]=111), the least signficant bit is always 0.

- In field-setting descriptions for VCOMSEL in Module Configuration Register (SDADC_MCR)
  - For the field setting 0, changed the description from "Negative input terminal is biased with VREFP/2 (half-scale bias)" to "Negative input terminal is biased with VREFN"
  - For the field setting 1, changed the description from "Negative input terminal is biased with VREFN" to "Negative input terminal is biased with VREFP/2 (half-scale bias)"
- In the table in the description of ANCHSEL in Channel Selection Register (SDADC_CSR)
  - For the case when MODE is 1 and VCOMSEL is 0, changed the "INM (negative terminal)" from VREFP/2 to VREFN
  - For the case when MODE is 1 and VCOMSEL is 1, changed the "INM (negative terminal)" from VREFN to VREFP/2

- Changes to SDADC_MCR register :
  - Reserved field containing bits 0-2 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bit 8 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bits 25-26 (most significant bit is numbered 0) is read-only zero (was R/W)
- Changes to SDADC_CSR register :
  - Reserved field containing bits 0-7 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bits 16-20 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bits 24-28 (most significant bit is numbered 0) is read-only zero (was R/W)
- Changes to SDADC_RKR register :
  - Reserved field containing bits 0-15 (most significant bit is numbered 0) is read-only zero (was R/W)
- Changes to SDADC_SFR register :
  - Reserved field containing bits 0-12 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bits 16-22 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bits 24-26 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Clarification: Added following note to DFFF field description: "Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to also clear the RSER[DFFDIRE] bit alongwith. Similarly if DMA or software is too slow in not reading the FIFO datawords resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both these action will ensure safe operation."
- Changes to SDADC_RSER register :
  - Reserved field containing bits 0-13 (most significant bit is numbered 0) is read-only zero (was R/W)
  - Reserved field containing bits 17-27 (most significant bit is numbered 0) is read-only zero (was R/W)
- Changes to SDADC_OSDR register :
  - Reserved field containing bits 0-23 (most significant bit is numbered 0) is read-only zero (was R/W)
- Changes to SDADC_STKR register :
  - Reserved field containing bits 0-15 (most significant bit is numbered 0) is read-only zero (was R/W)
- Changes to SDADC_CDR register :
  - Register access changed to Read-only
  - Reserved field containing bits 0-15 (most significant bit is numbered 0) is read-only zero (was R/W)

- Status Flag Register (SDADC_SFR) :
  - Added detail to WTHH field describing how field is cleared by DMA transfer
  - Added detail to WTHL field describing how field is cleared by DMA transfer
  - Clarification: Rephrased description of condition after which CDVF field is set. Field is automatically set when an internal timer counts a number of output clock fd clock cycles specified by the value of SDADC_OSDR[OSD].
- Data conversion :
  - Complete rewrite of first part of section up to the paragraph that describes wraparound control, to clarify the description of the output settling time required after startup. To account for delays related to latency and output settling time an internal timer counts down from a start value determined by the value of the SDADC_OSDR[OSD] field.

- External signal description :

- Replaced: "At the chip integration level, some of the digital and analog signals might share pins or may not be available external to the chip" replaced with "See the chip-specific details about SDADC signals and package pinouts for information about signal muxing and availability external to the chip."
- Gain calibration support :
  - Formatting changes: Binary number format changed to general form of 0000_1111_0000_1111b (was in form 0b0000_1111_0000_1111)
  - Formatting changes: Sequence of three notes changed to numbered list under one note heading (no technical change to content)
- Offset calibration support :
  - Correction: For step configuring SDADC_CSR[ANCHSEL] field value to '101', "single ended mode with negative input=VDD_HV_ADR_D–VSS_HV_ADR_D" changed to "single ended mode with negative input = $(V_{DD\_HV\_ADR\_D} – V_{SS\_HV\_ADR\_D}) / 2$"
  - Clarification: Fields are referred to using the notation, SDADC_*register_name*[*field_name*], e.g., SDADC_CSR[ANCHSEL]
  - Formatting change: Binary number format changed to general form of 0000_1111_0000_1111b (was in form 0b0000_1111_0000_1111)

- Throughout: Changed hexadecimal numbers prefixed with "0x" to "h" suffix
- Features :
  - Clarification: The programmable FIFO structure can store up to 16 datawords
  - Correction: Interrupt/DMA request generation condition for a data buffer threshold event occurs when the data buffer is *above* the threshold (previously stated "Data buffer *at or above* threshold")
- Differential input mode :
  - Clarification: Fields are referred to using the notation, SDADC_*register_name*[*field_name*], e.g., SDADC_CSR[ANCHSEL]
  - Clarification: Differential input mode is entered when the SDADC_MCR[MODE] field is set to '0' and the SDADC_MCR[EN] field is set to '1' (previously stated: "After system reset exit, this is the default mode of operation if SDADC is enabled by asserting MCR[EN]. This mode is entered by negating MCR[MODE]")
- Single-ended input mode :
  - Clarification: Fields are referred to using the notation, SDADC_*register_name*[*field_name*], e.g., SDADC_CSR[ANCHSEL]
  - Clarification: Single-ended input mode is entered by setting the SDADC_MCR[MODE] and SDADC_MCR[EN] fields to '1'. (previously stated: "Single-ended input mode is entered by asserting MCR[MODE]")
- FIFO Control Register (SDADC_FCR) :
  - Correction: The SDACD_FCR[FTHLD] field is 4 bits wide (bits: 20–23)
  - Correction (in the FTHLD field description): When the number of datawords in the data FIFO is greater than the value in FTHLD field, the FIFO full event is flagged (previously stated FIFO full event occurred when the number of datawords in the FIFO is *equal to or greater* than the value in FTHLD field)
- Data conversion :
  - Correction: Fixed typo ("SDADC" in following note was "SDADCD"):

    Care should be taken while programming the TRIGEN/ TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.

- Output Settling Delay Register (SDADC_OSDR)
  - In OSD field description, added note "Refer to the tsettling and tLATENCY parameters in the device datasheet for the minimum allowed output settling delay values."
- Data conversion step
  - Added sentence “Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.“ into the note.

A.41 SARADC changes

- Start of normal conversion :

• Following note added: "An appropriate gap should be maintained between any two consecutive triggers such that next trigger is received only after the completion of ongoing conversion. This gap value can calculated as a sum of total conversion time as programmed using the CTR0-3 registers for each channel in the chain." • Injected channel conversion : • Following note added: "An appropriate gap should be maintained between any two consecutive injected triggers such that next injected trigger is received only after the completion of ongoing injected conversion. This gap value can calculated as a sum of total conversion time as programmed using CTR0-3 registers for each channel in the injected chain."
• Main Configuration Register (SARADC_MCR) • Updated MCR reset value to 0x0000_8001 from 0x0000_0001.

A.42 Temperature Sensor changes

• In Temperature threshold detection (digital output generation) section, changed "..... if any of the three specified temperature thresholds are crossed" to "..... if any of the specified temperature thresholds are crossed"
• Replaced terms "bandgap voltage" and “reference voltage” with "bandgap reference voltage" across chapter • In Calculating device temperature, “$T_{TSENS\_CODE}(T)$” corrected to “$V_{TSENS\_CODE}(T)$”.
• In Temperature threshold detection (digital output generation) section, removed “or 165°C” in reference to hot insertion test temperature.
• In Temperature formula, removed the "Calibration Points" figure and its introductory sentence.

A.43 STM module changes

• In STM Control Register (STM_CR), added the note "The reset value is implementation specific. See the Configuration information." at the beginning of the section.
• In STM Control Register (STM_CR) description, removed the note "The reset value is implementation specific. See the Configuration information."
• In Functional description • Incorporated minor edits
• For reserved fields of following registers, added text to each field description: "This field is reserved." • STM Control Register (STM_CR) • STM Channel Control Register (STM_CCR*n*) • STM Channel Interrupt Register (STM_CIR*n*)

A.44 SWT module changes

• In Introduction, changed "timer" to "window watchdog timer".
• In SWT Control Register (SWT_CR), added the note "The reset value for the SWT_CR is implementation specific. See the configuration information." at the beginning of the section. • In SWT Time-out Register (SWT_TO), added the note "The reset value for this register is implementation specific. See the chip configuration section." at the beginning of the section.
• Memory Map and Registers: incorporated various edits • In SWT Control Register (SWT_CR) description, revised note:

• From: "The reset value for the SWT_CR is implementation specific. See the configuration information." • To: "The reset value of this register is implementation specific. See the chip-specific SWT information." • In SWT Time-out Register (SWT_TO) description, revised note: • From: "The reset value for this register is implementation specific. See the chip configuration section." • To: "The reset value of this register is implementation specific. See the chip-specific SWT information."
• In SWT_CR register • In section title, changed SWR to SWT • Edited text in section, including in footnote
• In Servicing operations incorporated minor editorial/formatting changes
• In SWT Control Register (SWT_CR) changed Reserved from read-only and always zero to read/write
• In SWT Service Register (SWT_SR) description of WSC field, changed beginning of second sentence from "If the SWT_CR[KEY] bit is set" to "If the SWT_CR[SMD] field is 01b" • In SWT Service Key Register (SWT_SK) description of SK field, changed beginning of second sentence from "If SWT_CR[KEY] is set" to "If SWT_CR[SMD] is 01b"

A.45 PIT module changes

• No substantial content changes

A.46 GTMINT changes

• Extensive revisions and improvements throughout the entire chapter.

A.47 GTM104 changes

• No substantial content changes

A.48 GTMDI changes

• Editorial changes. • Added About this module. • In External signal description, deleted the "Message data out (MDO)" and "Message start/end out (MSEO[1:0])" sections. • Added GTM-IP Debug / Halt mode description. • Revised Debug enable logic.
• In Device Identity register (DID), updated MPC5777M DID_PIN. • In Device Identity register (DID), added values of DID_PRN, DID_DC. • Revised the note after TIM-TOM-ATOM selection and control logic.

A.49 CAN Subsystem chapter changes

• In Core Release Register (M_CAN_CREL), added the note "The coding of revisions depends on the module version used in the device." at the beginning of the section. • In Core Release Register (M_TTCAN_CREL), added the note "The coding of revisions depends on the module version used in the device." at the beginning of the section.
• In Fast Bit Timing and Prescaler Register (M_TTCAN_FBTP), updated the bit field width for TDCO, FBRP bits and updated bit offset of TDC bit.

A.50 DSPI changes

• In DSPI DSI Configuration Register 1 (DSPI_DSICR1) :
  • Updated bit-width of TSBCNT bit from [2:7] to [3:7].
  • Updated bit 3 as Reserved.
 	
• In DSPI Status Register (DSPI_SR) : Updated reset value from 0xDEAD_BEEF to 0x0201_0000h.

• Changed access for all the reserved bit fields in the DSICR0 register to R/W. • Updated DSI serialization
• In section, Continuous Serial Communications Clock, Updated sentence, “Enabling this bit generates the Continuous Serial Communications Clock regardless of the MCR[HALT] bit status” to “Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low".

• All reference to DSPI DSI 64bit Mode removed from the RM as this mode is not supported by the device.
  • In Memory Map and register definition
  • DSPI_CTAR0: Updated FMSZ field description to remove DSI64 support.
  • DSPI_CTAR_Slave: Removed FMSZ5 bitfield and Updated FMSZ field description to remove DSI64 support.
  • DSPI_DSICR0: Removed FMSZ5 bitfield.
  • DSPI_SDR0: Updated description to remove DSI64 support.
  • DSPI_ADSR0, COMPR0, DDR0, SSR0, DIMR0, DPIR0: Updated description to remove DSI64 support.
  • Updated bitfield description DSPI_DSICR1[TRGPRD.]
  • Updated bitfield description DSPI_DSICR1[DSE1] to remove DSI64 support.
  • Deleted DSPI_DSICR1[DSI64E] bitfield.
  • Deleted registers, DSPI_SDR1, DSPI_ADSR1, DSPI_SSR1, DSPI_COMPR1, DSPI_DDR1, DSPI_DIMR1, DSPI_DPIR1.
  • Revised following sections to remove DSI64 Support
  • Timed Serial Bus (TSB)
  • MSC Dual Receiver Support with PCS Switch Over
  • Interleaved TSB (ITSB) Mode
  • Features
  • DSI serialization
  • DSI deserialization
  • DSI configuration
  • Deserialized Data Match Interrupt or DMA Request
  • Deserial Serial Interface (DSI) configuration
  • CSI deserialization
  • CSI serialization
  • Change in data control
• Hidden unimplemented PISR0-7 registers

A.51 Zipwire module changes

• Write performance   • Updated the 'SIPI to write an address' time to 310 ns.

• In Introduction :
  • Updated the paragraph text from "In streaming mode Zipwire has a transfer rate of approximately 32 MB/second" to "In streaming mode Zipwire has a high transfer rate".
• In Read performance :
  • Updated the paragraph text from "64-bit Read response takes 278 ns" to 64-bit Read response takes 278 ns (=89 x 3.13 ns)".

• Editorial changes
• In Zipwire interconnections : Updated bullet from "Tx and Rx connections to the pads via the SIUL and the MSCR registers" to "Tx and Rx configuration is controlled by LFAST Control registers". • Removed "LFAST clocking" section.
• In Architecture : Removed paragraph text "The LVDS pads are shared with other I/O and GPIO.".

A.52 SIPI module changes

• Reserved
  • Added note (Important), "Always write the default value to this field." to the Reserved field description (bit 1).
  • Changed bit 1 to Read/Write from Read Only.

• In In ARR register added a note "This register is writeable only when SIPI_MCR[INIT] = 1." Also removed sentence, "This register can be read or written anytime." from register description.
• In CCR0[WRT], removed a note "Only transfers with channel 2 can write 1 to this bit. All other channels forces SIPI_CCRn[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming.” from the WRT bit field of CCR0-CCR3 registers. • Added bit field brief description of WRT bit in CCR2 register.

A.53 LFAST module changes

• Editorial changes.

• Removed reference to 312/320 MHz and referred to as high data rate. Updated in:
  • External signals section.
  • LFAST operating data rates section.
  • Features section.
  • Auto correlation section.
  • Figure 52-6 figure.
  • LFAST Speed Control Register (LFAST_SCR) : Updated description for bit-value 1 of RDR and TDR bit.
• Editorial changes.
• Removed table from LFAST operating data rates section.
• Changed the base address of LFAST register set from 1h to 0h.

• LVRXOP bit in LFAST LVDS Control Register (LFAST_LCR) is split into 3 individual bits: • LVRXOP[2] changed to LVRXOP_TR.

• LVRXOP[1] changed to Reserved. • LVRXOP[0] changed to LVRXOP_BR. • Updated the access value of LVLPEN bit to Read/Write in LFAST LVDS Control Register (LFAST_LCR).
• Updated the description of [LVCKSS] bit in LFAST LVDS Control Register (LFAST_LCR).

A.54 FEC module changes

• In MII Management Frame Register (FEC_MMFR), added the note "Does not reset" at the beginning of the section.
• In Memory map and register definition, changed the base address appearing above each register figure (is 0h).

• In Features, changed the footnote to "For interface selection options, see the chip-specific FEC information."
• In Interface options, changed the footnote to "For interface selection options, see the chip-specific FEC information."
• In MII Management Frame Register (FEC_MMFR), changed the reset value to X (indicating that the reset value is undefined, as mentioned in the description).
• In Memory map and register definition :
  • Added a new section, Using the RMON and IEEE registers.
  • Revised all the RMON and IEEE register definitions to match the information in this new section.

• In MII Management Frame Register (FEC_MMFR), deleted the text "Does not reset" associated with the reset value, and ensured that the register reset value information is correctly presented.
• In Physical Address High Register and Type Field (FEC_PAUR), changed the reset value of the TYPE field (was 0000h, is 8808h) to match what is already written in the register and field description. • In Opcode/Pause Duration (FEC_OPD), changed the reset value of the PAUSE_DUR field (was 0000h, is undefined at reset) to match what is already written in the register and field description.
• Added Clocking requirements.
• In Frames received with CRC error (FEC_IEEE_R_CRC), ensured that the reset value is shown as Undefined.

A.55 FlexRay module changes

• Glossary : Minor editorial update.
• Minor editorial updates in few sections. • Features : Clarified features for SECDED ECC.
• Changed "Memory Content Error Detection" section title to "Memory Content Fault Detection". • Changed "Memory Error Injection" section title to "Memory Fault Injection". • Changed "Memory Error Injection out of POC:default config" section title to "Memory Fault Injection out of POC:default config" and changed references of "error" to "fault" in this section.

• Added the note "After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section)." at the beginning of the following sections:
  • Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*)
  • Message Buffer Frame ID Register (FR_MBFIDR*n*)
  • Message Buffer Index Register (FR_MBIDXR*n*)
  • Message Buffer Data Field Offset Register (FR_MBDOR*n*)
  • LRAM ECC Error Test Register (FR_LEETR*n*)

• Memory map and register definition : Editorial updates.
• Slot Status Counter Condition Register (FR_SSCCR) : In FR_SSCCR[STATUSMASK[3:0]] bit field name, removed the text [3:0].
• StopWatch Count Register (FR_STPWR) : corrected the width of FR_STPWR register and FR_STPWR[STPW] from 16-bits to 32-bits

• FR_MCR[CHB]: In FlexRay channel selection table, Dual channel Device Modes CHB=1 and CHA=0 settings are updated as "reserved"
• In Features, added "Dual Channel Support" feature.

A.56 I2C changes

• In I2C Bus Data I/O Register (I2C_IBDR), added the note "When the I^2C is configured ... should be different".
• Editorial changes.

• Editorial changes.
• In I2C Bus Data I/O Register (I2C_IBDR), deleted the note "When the I^2C is configured in master mode ... should be different".
• In Figure 55-13, changed "Set TXAK =1" to "Set NOACK = 1".
• In Generating START, changed IBVR to IBCR.
• Revised Transmit/receive sequence to show the correct procedures for all four cases.
• In Exiting DMA mode, system requirement considerations :
  • Revised the information about DMA channel linking and DMA scatter/gather process.
  • Added an example for a 150 MHz system.

• In Memory map and register definition, added register access level information (indicating that registers can be accessed in supervisor mode only).
• Editorial changes. • In I2C Bus Interrupt Config Register (I2C_IBIC), added the BYTERXIE field. • In Table 55-15 and Table 55-16, deleted the first step ("Use the IBFD register to select...") and adjusted the remaining step letters accordingly. • In Loss of arbitration, changed "generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed" to "generate an interrupt to CPU, set the IBAL to indicate that the attempt to engage the bus is failed, and not set the TCF due to the loss of data during arbitration."
• In I2C Bus Frequency Divider Register (I2C_IBFD), revised the IBC field description to show the correct bit encoding. • In I2C Bus Interrupt Config Register (I2C_IBIC), added "To program BIIE = 1, you must ensure that IBCR[MDIS] = 0." • In Table 55-11, changed the "SCL Hold (stop)" entry for IBC = 6F (was 96, is 962).
• In Register accessibility, added "Address location 0x0007 is a reserved location, but access to this location will not generate any bus error."
• In Figure 55-1, changed the SCL and SDA lines to appear bidirectional. • In Arbitration procedure and Loss of arbitration, changed "slave receive mode" to "slave mode".

A.57 PSI5 module changes

• Updated DSR_RST fields of CHn_DOBCR registers with current value of DEFAULT_SYNC field. • Changed the reset value of Global Control Register (PSI5_GCR) register from 0000 to 0001. • In the PSI5_GCR register GLOBAL_DISABLE_REQ field description, removed sentence “When PSI5_PDIS_RST = 0 then the reset value is "0"(PSI5 module is enabled after reset). • Editorial change.
• A note added in DMA support section: DMA master ID replication mode is not allowed together with PSI5. • Removed "Clock and Reset" section.
• Changed S3SBR+ to S3SBR in all channels in SnEBR resgiters' description.
• Added T-bit handling and all its sub-sections.
• In Features section, removed "DMA transfer through bullet list" feature from the features' list. • In Operating modes section, updated the "Operating Modes" diagram.

• In External signal description section, removed the duplicate footnote from the "PSI5 Port List" table. • Changed the location of table footnote to page footnote. It now appears in the description of PSI5_GCR register. • Updated the description of NDS field of PSI5_CHn_NDSR registers. • Updated the description of DSR field of PSI5_CHn_DSRL and PSI5_CHn_DSRH registers.

• In Data Output Block Configuration Register (PSI5_CH0_DOBCR) register section,
  • Deleted duplicate footnotes from "DOBCR bit configurations and the related output states" table.
• In Data Output Block Configuration Register (PSI5_CH1_DOBCR) register section,
  • Deleted duplicate footnotes from "DOBCR bit configurations and the related output states" table.
• In Data Output Block Configuration Register (PSI5_CH2_DOBCR) register section,
  • Deleted duplicate footnotes from "DOBCR bit configurations and the related output states" table.

• In PSI5_CHn_DSR[IS_DMA_PM_DS_FIFO_FULL] description, changed "the DMA request ipd_psi5_dma_req_pm_ds" to "DMA request"
• In PSI5_CHn_GICR description, reorganized second half of sentence into list format

• In each CHn_DOBCR register, in the "CMD_Type" table:
  • Made formatting updates in the heading row
  • Combined redundant footnotes and edited footnote text
• In CH0_DOBCR register, removed "DPR" after "State 5" in "DOBCR bit configurations and the related output states" table.

A.58 PSI5-S module changes

-
• Editorial change.
• In PSI5-S Watchdog Operation corrected cross-references to register descriptions of WD_CFGR_CH and WDGTSSR
• Updated field width of IFD field in PSI5-S UART Tx Idle Delay Time Register (PSI5S_PTD) register to 4-bit, in place of 5-bit earlier.

• Added PSI5-S ECU to Sensor Direct Command Write register (PSI5S_DIRCMD).
• Moved position of DIRCMD_RDY field in PSI5-S Global Status Register (PSI5S_GLSR) from 12 to 11.
• Removed the following registers:
  • WD_DDTRIG_CLKDIV0
  • WD_DDTRIG_CLKDIV1
  • WD_DDTRIG_CLKDIV2
  • WD_DDTRIG_CLKDIV3

A.59 SRX/SENT module changes

• In Global Control Register (SRX_GBL_CTRL) description:
  • Changed FMDUIE field to Reserved
  • Changed SMDUIE field to Reserved

• In Global Status Register (SRX_GBL_STATUS) description, changed abbreviated name of Fast Message DMA Underflow field from FDMU to FMDU
• Minor editorial updates • In section Low power modes added a note to show that the message read registers will appear to lose their contents in certain conditions. • In section Initialization sequence added a note to explain that NUM_EDGES_ERR status flag is getting set randomly at the falling edge of the first Calibration pulse (start of status and communication nibble) which comes after an IDLE phase, even without the occurence of the event related to this flag and still the receiver is able to receive the message. Because of this other error status flags may get masked.

- In Fast Message Ready Status Register (SRX_FMSG_RDY) added a note in SRXx_FMSG_RDY register to show that under certain conditions, the Single Edge Nibble Transmission (SENT) Receiver (SRX) stalls and the Fast Message Data Ready bit for the SENT channel (FMSG_RDY[F_RDYn]) will no longer get set to indicate that a fast message is available. Reads of any of the fast message registers by the MCU core will stall and not complete.
- In section Adjustment for variation in sensor (Tx) clock note added as a workaround for Length of Calibration pulse 25% higher
- In section CRC check note added as a workaround for SENT module that creates errors when the actual calibration pulse is shorter than nominal value
- In section Pause pulse diagnostic added details as a workaround for unexpected behavior of SENT Calibration equal to pause pulse
- Deleted heading "Register Classification for Safety"

- In table “Table 58-2" ID [15:12] changed to Data Field [15:12]

- In Features changed sentence from "Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift and jitter in the Transmitter and Receiver clocks per channel" to "Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel".
- In Slow Serial Message Ready Status Register (SRX_SMSG_RDY) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver".
- In Fast Message Ready Status Register (SRX_FMSG_RDY) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver".
- In Initialization sequence, Note removed as it was taken care of via ERR007425
- In Fast Message DMA Control Register (SRX_FDMA_CTRL) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver".
- In Slow Serial Message DMA Control Register (SRX_SDMA_CTRL) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver".
- In Fast Message Ready Interrupt Control Register (SRX_FRDY_IE), changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver".
- In Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver".
- In Input programmable filter updated value of FIL_CNT for the example shown and updated the diagram
- In CRC check removed Note
- In Fast Message Ready Status Register (SRX_FMSG_RDY) removed a Note
- In Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) added a Note to the field CM_PRSC
- In Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) changed sentence from "The generated Receiver Clock is compensated for variations, clock drift and jitter that might occur in the Sensor Tx Clock" to "The generated Receiver Clock is compensated for variations and clock drift that might occur in the Sensor Tx Clock".
- In Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS) changed SMSG_OFLW bitfield description.
- In Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG) changed description of field values of bitfield SRX_CHn_CONFIG[FCRC_TYPE] and SRX_CHn_CONFIG[SCRC_TYPE].
- In Adjustment for variation in sensor (Tx) clock deleted note that starts with "At ambient ...".
- In Receiver diagnostics changed bullet point from "Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or XOR-based" to "Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or Recommended.".
- In Nibble value check changed intoductory sentence from "As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 ticks:" to "As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 nibble values(11.5 ticks to 27.5 ticks):".
- In CRC check changed text in introductory paragraph from "XOR-based CRC implementation" to "new recommended CRC implementation".
- In Pause pulse diagnostic update changed table caption and deleted FMSG_OFLW row.
- In High frequency receiver clock (protocol clock) requirements modified entire content within this heading.

- In Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL), changed access of CM_PRSC field from read/write to read-only.

- In Time stamp logic changed text:
  - Was: "For example, if the high frequency receiver clock is 59 MHz, the required prescaler value to generate a time stamp clock of 1 µs resolution is 60"
  - Now: "For example, if the high frequency receiver clock is 60 MHz, the required prescaler value is 59 to generate a time stamp clock of 1 µs resolution"

• In DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) changed field name SRX_DMA_SMSG_BIT3[ID7_4_D3_0] to SRX_DMA_SMSG_BIT3[ID7_4_ID3_0].
• Minor editorial changes.

A.60 LINFlexD module changes

• In the description of LINFlexD_UARTSR[RMB], changed LINESR to LINSR
• In the LINFlexD_LINCR1 register description, added note: "When accessing the LINFlexD_LINCR1 register, each reserved bit should be written to its original reset value." • In the LINFlexD_UARTCR register description, added note: "When accessing the LINFlexD_UARTCR register, reserved bits should always be written to zero." • In the LINFlexD_LINCR2 register description, added note: "When accessing the LINFlexD_LINCR2 register, each reserved bit should be written to its original reset value."
• In UART mode section, added "13-bit frames" to second bullet item.
• Updated the reset value note: "When slave = 0, this field always reads ‘0’.." and the bit field description: "If generic slave.." for IOPE bit of LINCR2 register. • Added notes to UARTCR and LINTCSR register descriptions • In section: Use cases and limitations, deleted list item, "When IPG_STOP is requested..."
• Updated the reset value note for HTO bit of LINTOCR register • Added note to RDFL_RFC and TDFL_TFC bit description of UARTCR register
• Added note to reset value of MME bit of LINCR1 register

• In Timeout error section
  • Changed ocr_1 to ocr_2 in headings: 'Master mode' and 'Slave mode'.
  • Changed OC1 to OC2 in figure: 'No response'
• In LINFlexD_UARTSR register, removed text from descriptions of DRFRFE and DTFTFF fields: "Register bit is read-only field."
• In LINFlexD_LINOCR register description, changed LINTOCR value to LINTCSR:CNT value
• In LINFlexD_LINTOCR register, clarified HTO footnote

• Editorial updates in the following sections:
  • Slave mode
  • Identifier filtering
  • Start detection and break delimiter detection in receiver
  • UART receiver
  • UART transmitter

• Clarified baud rate calculation equation/formula to have two cases: ROSE=0 and ROSE=1 in section Baud rate generation.
• Added list item, "When IPG_STOP is requested..." in section: Use cases and limitations which was previously deleted.
• Deleted the note: "When the MODE bit is 0, any activity on the transmit ..." written in the register description of LINTCSR. • Updated the reset value note written in the IOPE bit reset value of register LINCR2.

• In section: Master mode, changed 'BDR[0:7]' to 'BDRL and BDRM'
 	
• In section: Memory map and register description,
  • Updated the note: "In Master mode, the registers IFCR0 – IFCR15...". Added table: "Offsets of the register from offsets 8C to 9C", below the note.
  • Minor editorial updates in register 'LINFlexD_LINCR1', bit field 'LASE'

• Changed "For proper functionality, OSR can only take values" to "Allowed values are:" in register 'LINFlexD_UARTCR', bit field 'OSR' • Removed phrase "Parity sent is Even" from the bit field description of bits 'WL0' and 'WL1' of register LINFlexD_UARTCR
• In section: LIN mode features and UART mode features : Updated the baud rate calculation relationship. • In section: Memory map and register description, • Removed reset value of register 'LINFlexD_LINSR' and added the following note to the bit reset value of bit field 'RDI: • "Reset value of RDI reflects the RX pin state" • Updated the bit field description of bit fields 'Reserved' and 'DTE' of register 'LINFlexD_DMATXE'. • Updated the bit field description of bits fields 'Reserved' and 'DRE' of register 'LINFlexD_DMARXE'.
• In section: Timeout error, • Changed 'At the end of the ID, ocr_2 is loaded with 36 (maximum possible response space)' to 'At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT]' • Changed 'ocr_2' to 'OC2'
• In section: Memory map and register description, • Removed sentence "The reset value is the 2Dh = 45 ... the LIN specification." from the description of 'HTO' bit of register 'LINTOCR' • In section: Interrupts, added note: "Rx interrupt due to ... LSIE is enabled." in the figure: 'Interrupt diagram'
• In section: Memory map and register description, • In register LINSR, • Changed the access of bit field 'LINS' to ROWO (was RO) • Added note 'The value of this bit field doesn't change ... LIN state event. to the bit field description. ' • In register, UARTSR, updated the bit field description of bit DRFRFE. • In register, UARTSR, updated the bit field description of bit DTFTFF: 'This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.' to 'This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set.' • In register LINTOCR, clarified the reset value note of bit field HTO.
• Updated section: Header error
• In section: References", removed note: 'FBRR will not be usable ... in UART mode.' • In section: Baud rate generation, updated sentence 'When reduced oversampling is enabled, LINFBRR is not used and LDIV contains only the integer part of LINIBRR.' to 'When reduced oversampling is enabled, LINFBRR should not be used and programmed to zero and LDIV contains only the integer part of LINIBRR.' • In section: Memory map and register description, • In register UARTCR, enhanced the bitfield description of bits WL0, WL1, PC0, and PC1.
• In section: Memory map and register description, • Updated the bitfield descriptions of: • Bitfields 'SLEEP' and 'INIT' of register 'LINCR1' • Bitfields 'DTU_PCETX' and 'TDFL_TFC' of register 'UARTCR' • Bitfields 'EN' and 'IFD' of register 'PTD' • Updated the register description of register 'PTD' • Reduced the width of bitfield 'IFD' from 5 to 4 of register 'PTD'
• Updated section: Baud rate generation
• In section: Memory map and register description, • In register 'LINESR' updated the bitfield description of 'OCF'
• Added section: LINFlexD Clock Tolerance

A.61 MC_RGM module changes

• Added MC_RGM_FERD[D_SOFT_FUNC] field.

- In Memory Map and Register Definition, Updated note, 'All registers will be accessible in "User mode"... only if SSCM_ERROR[RAE] is set'.
- Editorial update to MC_RGM_FERD[D_TSR_FUNC] field description.
- Added note to MC_RGM_FES, 'When ESR0 gate is disabled via ... ESR0 reset has also occurred.'

A.62 BAF module changes

- Download start address, VLE bit and code size
  - Removed the paragraph "The NO_ECHO bit is used to indicate..."
  - Table Table 61-19
    - Replaced the row in table that has VLE at bit0, and now have a CODE_LENGTH of 15 bits (CODE_LENGTH[30:16])and no NO_ECHO bit.
- Protocol summary
  - Table Table 61-21
    - Removed the contents of row 'Protocol steps' 4, column 'BAF response message' and replaced with '—'. NO_ECHO bit is not used.

- BAF image header
  - Removed the device specific 'BAF version number' address, replacing it with text, "BAF version number". Customer is referred to the 'BAF memory map' table in the configuration section.
- BAF image version
  - Removed the device specific BAF start address, and referred customers to the "BAF memory map" table in the configuration section.
- Optionally perform serial boot
  - Switched the output of the "Password Valid" decision block, 'no' to 'Static mode' and 'yes' to 'Receive START address' in Figure 61-3

- Bullet point describing "Serial Boot" in Boot modes section is updated.
- Flow of control image in Flow of control section is updated.
- Table 61-4 is updated for clarity.
- Note added in Soft DCF clients section and Table 61-5 is updated.
- Note added in Security Watchdog CPU select (SEC_SWT_CPU) section.
- Caution is added in BAF DCF client - DATAX section.
- New section added - CALIBRATION_PD_DELAY_ENABLE – BAF calibration PD delay enable DCF client.
- In second paragraph of Production disable activation protocol and implementation section, modified the value to enable jitter.
- New section added - TDM Diary Operation.
- Download start address, VLE bit and code size section is updated as:
  - The statement mentioning about 3 LSB bits in download address is updated in the paragraph starting as "The start address...".
- Protocol summary section is updated as:
  - Table 61-21 is updated.
  - Row for protocol step 2 in Table 61-22 is updated.
- Resources is updated as:
  - Bullet point mentioning PFLASH is updated.
  - New bullet point - FCCU is added.
  - New bullet point - TDM is added.

- Added "or JDP/FSL Production" to the first paragraph of Optionally perform serial boot section.
- Updated Download start address, VLE bit and code size section:
  - cross link from 1st paragraph changed to Table 61-19.
- Changed figure title in Implementation details of BAF supplied remap function section to "BAF flowchart default implementation".

- Added a DCF client, IVPR2_CPU2 in Table 61-5
- Added a new section - IVPR2_CPU2 - BAF set IVPR2 for CPU2 DCF client

- Updated Introduction section.
- Added Caution note in BAF DCF client - DATAX section.

• Added new section Clock Settings under Serial boot configuration. • Updated reference to baud rate frequency in LINFlexD configuration section. • Updated the baud rate frequency in the paragraph under section - Baud rates and in Table 61-20.
• Updated Figure 61-1. • Added a new row in Table 61-17 for "Production Disable flash Driver SRAM area".

A.63 SSCM module changes

• Life Cycle   • Added NOTE at end of section "When programming Life Cycle, ensure that..."
• ECC error monitoring   • Reformatted the list of possible error reactions to a bullet list for better readability.

• SSCM System Status (SSCM_STATUS)
  • Removed the NOTE "Reset value depends on the device status after leaving reset." from the BMODE field description.
• BAF configuration
  • Updated the paragraph "In BAF mode..."

• In the SSCM memory map table, changed the access for LCSTAT register from "R/W" To "R" • For LCSTAT register reserved bits 22/23, added footnote ("Reset Value is...") and changed reset value to undefined
• Modified footnote in Table 62-5

• In Life Cycle Status Register (SSCM_LCSTAT) description of LC field settings
  • Changed "FAIL_ANALYSIS" to "Failure Analysis"
  • Changed "Undefined" to "Reserved"
  • Changed "PROD_OEM" to "OEM Production"
  • Changed "CUST_DELIV" to "Customer Delivery"
  • Changed "JDP_PROD" to "JDP Production"
  • Changed "IN_FIELD" to "In Field"

• In Table 62-5 updated the abbreviated names of LC slot table to match the updated ones in SSCM_LCSTAT[LC].
• Updated Features list
• Added PWCMPH register. • Added PWCMPL register.
• In SSCM System Status (SSCM_STATUS) register updated the reset value and description of bit 6 (Reserved)
• In CER removed "(with PARITY enabled)" from the bit description.

• In SSCM System Status (SSCM_STATUS) register's CER field description, changed first sentence and added sentence:
  • was: "This field indicates that the SSCM has detected a configuration error during reset while loading DCF clients."
  • now: "This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error."

A.64 PMC_DIG Changes

• Introduction   • Editorial updates. • Pending Gauge Status Register (PMCDIG_GR_P)

- Added "This bit is indeterminate after reset..." to the VD10_A, VD9_IF, VD9_IJ, MD9_IM, VD9_F, and VD9_C bitfield descriptions.
- Added "This bit is indeterminate after reset..." to the VD10_F and VD9_O bitfield descriptions.

- Analog PMC interface
  - Removed the text " (both active high and active low, but only active low are used)" from the paragraph "The signals from the PMC..."
  - Replaced the sentence "This time can vary from 5 µsec to 11.4 µsec." with "This time can be ≥ 5 µsec."
- Temperature Sensor interface logic
  - Editorial updates.
  - Removed the text "The test interface is discussed in the test section."
- Standby RAM regulator interface logic
  - Removed the paragraphs "Another control signal, vrefsel, is controlled..." and "There are a few test signals that are also used..."
- Power On Reset (MC_RGM phase gates)
  - Added "(and other system time counts)" to the paragraph "During the POR phase 3 exit, the..."
  - Editorial updates.
- Flash memory interface (DCF client usage)
  - Removed the sentence "Please see the Reset Event Enable (REE_TD) register for more information regarding REE/RES bits loading from flash."
  - Updated the content of the paragraph "These trim bits that are loaded..."
  - Editorial updates.
- Other module interfaces
  - Added "LVD295_A" to the LVD bullet list.
- Interrupt Enable Pending Register (PMCDIG_IE_P)
  - Added NOTE "Special care must be taken with..." to register description.
- Reset Event Select Registerr (PMCDIG_RES_VD8)
  - Added description details for the HVD8_F bitfield.
- LVD270 Event Pending Register (PMCDIG_EPR_VD9)
  - Added the NOTE "These bits are cleared out of initial..." to the register description.
  - Added the paragraph "This field is indeterminate..." to the LVD9_O, LVD9_IF, LVD9_IJ, LVD9_IM, LVD9_F, and LVD9_C field descriptions.
- LVD295 Event Pending Register (PMCDIG_EPR_VD10)
  - Added the NOTE "These bits are cleared out of initial..." to the register description.
  - Added the paragraph "This field is indeterminate..." to the LVD10_F and LVD10_A field descriptions.
- Temperature Sensor Configuration Register (PMCDIG_CTL_TD)
  - Added the text "For instance, 111b (-7) could be written to the..." to the end of the second paragraph in the register description.
  - Added NOTE "Care must be taken to include a timeout..." to the register description.
  - Added "This signal behavior has changed. The..." to the DOUT_EN field description.
- Voltage Detect User Mode Test Register (PMCDIG_VD_UTST)
  - Updated the register description.
- Module Control Register (PMCDIG_MCR)
  - Updated the register description.
  - Replaced the ADC_CHSEL_EN field with a Reserved field.

- In last paragraph of Other module interfaces corrected title of referenced document: MPC5700 Family Microcontroller Security Reference Manual

- In the Analog PMC Interface section, added a new paragraph about POR initialization.
- In the Temperature Sensor interface logic section, added a new paragraph describing the entry for Temp Sensor User Trim Adjust registers to prevent accidental resets.
- In the Temperature Sensor interface logic section, added a note at the end of section to describe the flag behavior and status bits.

A.65 MC_PCU module changes

- No substantial content changes

• In Memory Map and Register Definition, Updated note, 'All registers will be accessible in "User mode"... only if SSCM_ERROR[RAE] is set'.

A.66 MC_ME module changes

• In the Register description section, added MC_ME_CADDR2 register.
• In Memory map and register definition section: Updated access of MC_ME_SAFE_MC[PWRLVL] field from RO to RW
• In the Invalid mode configuration interrupt section: Removed the reference to ME_CCTLn and changed ME_CTL to ME_MCTL. • Removed note from CCTLx registers that stated, "A write access to CCTL register during MC_GS[MTRANS]=1, will result in the ICONF_CC flag in the ME_IS register being asserted." • Added note to the DBG_MODE bit in the MC_ME_DMTS register. • Added arrow in MC_ME Mode Diagram from TEST to SAFE. • Changed the description for the DBG_MODE field in the MC_ME_DMTS register. Changed all instances of the DBG_F field to Reserved in the MC_ME_PCTLx registers.
• Added note about the RMC bit to the CADDR4 register.
• In Memory Map and Register Definition, Updated note, 'All registers will be accessible in "User mode"... only if SSCM_ERROR[RAE] is set'.

A.67 Core Debug Support changes

• No substantial content changes

A.68 Core Debug Support changes

• In Hardware Debug Facilities, changed "Standard 1149.1" to "Standard IEEE 1149.1".
• Editorial changes. • In description of DBCR4[DVC1C] in Table 67-5 defined the field value corresponding to "Inverted polarity DVC1 operation" as 1
• Added the core name (e200z710n3) throughout the chapter.

A.69 DCI changes

• In "Software enabled debug and calibration mode" section, minor edit and removed second paragraph of NOTE. • Updated references to IEEE 1149.1 and IEEE 1149.7.

A.70 JTAGC module changes

- No substantial content changes

A.71 CJTAG changes

- Editorial changes.
- In Overview, removed references to SScan.

A.72 JDC changes

- Changed reset description of memory mapped registers to say registers are reset by system destructive reset.
- In Overview, removed sentence "The JTAG accessible registers are provided through an instance of the existing JTAGC module".
- In Memory mapped registers sections, changed base addresses of each register from "FFF3_C000h base" to "0h base".

A.73 SPU changes

- Added State1 offset in L2nSEL0 register.

A.74 JTAGM changes

- In the "Memory mapped registers" section, changed the base address appearing above each register figure (is 0h).

A.75 Zipwire_debug information changes

- No substantial content changes

A.76 SIPI_debug changes

• No substantial content changes

A.77 LFAST_debug changes

• No substantial content changes

A.78 DTS changes

• In the "Memory mapped registers" section, changed the base address appearing above each register figure (is 0h).

A.79 NAR changes

• No substantial content changes

A.80 Core Nexus Module changes

A.81 Core Nexus Module changes

• In Feature list, added bullet item "Indicates to the FCCU whether it becomes active during functional mode."
• Editorial changes.
• Corrected broken link.
• Replaced nex_evto_b with evto_b throughout chapter.
• Added missing TSTAMP sub-row to each Message Name row in Table 80-2 table. • Data Acquisition Trace event : in figure, corrected "TCODE(011011)" to "TCODE(000111)". • Added Timestamp field (TSTAMP). • Data Trace Messaging (DTM) : Changed "in one of three ways." to "in one of the following ways."
• Added the core name (e200z710n3) throughout the chapter.

A.82 NXMC module changes

• Updated Data Trace Size (DSZ) encodings TCODE value

A.83 CRC module changes

• Changed "CPU" to "DMA" in "Transmission phase 3" figure (bottom). • Added note to Standard features bullet regarding Ethernet CRC. • Updated CFG[SWAP_BITWISE] field description for value 1 from "Perform bit-wise swap on CRC_INP input data internally for CRC-16 polynomial calculations.", to "Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.".

• Removed references to support CRC-8-H2F Autosar polynomial and its impact from:
  • Main features section : Updated the bullet from "Four hard-wired polynomials..." to "Three hard-wired polynomials..."
  • Standard features section
  • Description of POLYG field in Configuration register (CRC_CFGn)
  • Functional description section

• Title corrected for section "Register programming".
• Added information about register access level at the end of each register description.
• Updated Main features section. • Editorial changes.
• In CRC_CFG*n*[SWAP], updated bit description text from "The swap operation is a bit-by-bit rotation of the content" to "The swap operation is a bit-by-bit swapping of the content".

A.84 MEMU changes

• Editorial changes.
• In Figure 83-4, changed "that is in error" to "that is false".
• In Memory map and register definition, added register access level information to all register descriptions.
• Previous errata e8145 integrated into this reference manual: In System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR), in the ERR_ADD field description, deleted "This register must be READ ONLY when its corresponding valid bit is not asserted." • Previous errata e8145 integrated into this reference manual: In Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR), in the ERR_ADD field description, deleted "This register must be READ ONLY when its corresponding valid bit is not asserted." • Previous errata e8145 integrated into this reference manual: In Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR), in the ERR_ADD field description, deleted "This register must be READ ONLY when its corresponding valid bit is not asserted."

• Editorial changes.
• In Introduction and Features :
  • Standardized the names of the error-reporting categories for consistency:
  • System RAM

• Peripheral RAM • Flash memory • Changed "see the Device Configuration chapter that describes how modules are configured and connected" to "see the chip-specific MEMU information." • In Design overview, changed "see the Device Configuration chapter that describes how modules are configured and connected" to "see the chip-specific MEMU information." • In Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*), in the BAD_BIT field description, changed "bit position in RAM" to "bit position in flash memory".
• Revised Figure 83-2 and Figure 83-3. • In System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR), added "This field must be read-only when its corresponding valid bit (SYS_RAM_UNCERR_STS[VLD]) = 0." to the ERR_ADD field description. • In Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR), added "This field must be read-only when its corresponding valid bit (PERIPH_RAM_UNCERR_STS[VLD]) = 0." to the ERR_ADD field description. • In Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR), added "This field must be read-only when its corresponding valid bit (FLASH_UNCERR_STS[VLD]) = 0." to the ERR_ADD field description.
• Revised Memory map and register definition so it shows 2 instances of the PERIPH_RAM_OFLW register.

A.85 IMA changes

A.86 FCCU module changes

• Added note to RF Enable Register (FCCU_RF_E*n*) description: "Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn)." • For Delta T Register (FCCU_DELTA_T) • Added note to description: "Reserved bits should always be written as all 0's." • Changed access for reserved field that was read-only and always 0 to simply read-only • Added text to description of following registers: "These registers can be written only when the FCCU is in CONFIG state." • IRQ Alarm Enable Register (FCCU_IRQ_ALARM_EN*n*) • NMI Enable Register (FCCU_NMI_EN*n*) • EOUT Signaling Enable Register (FCCU_EOUT_SIG_EN*n*)
• Changed the value for Delta_T from 10 ms to 16.67 ms in FCCU_DELTA_T register and also in Introduction Section. • Added note to the FCCU_NCF_EN register description: Any fault that is enabled should be programmed to take some action.
• Changed bit definition for PS bit of the FCCU_CFG register = 0 from "0 - eout[1] active high, eout[0] active high" to "0 eout[1] active high, eout[0] active low" • Changed bit definition for PS bit of the FCCU_CFG register = 1 from "1 - eout[1] active low, eout[0] active low" to "1 eout[1] active low, eout[0] active high"
• Introduction section - Deleted word "redundant" in first line. • Introduction section - In bullet item "The FCCU's configuration is lockable in two ways, deleted "in two ways" • Introduction section - Added footnote to bullet item "After power-on the error out pins have high impedance." Foot note reads "Actual value depends on the design settings at pad level." • Introduction section - In last paragraph, changed "specific FCCU state" to "CONFIG state" • Acronyms section - Added WKPU entry. • Acronyms section - Deleted NVM.

- Main Features section - Reworded first two bullet items "1 to 128 critical/unrecoverable ...." and "1 to 128 non-critical/recoverable ...."
- Fault priority scheme and nesing section - Changed "NCT timer" to "NCF timer"
- NMI/WKPU interface section - Deleted first paragraph beginning with "The wake-up interface includes the following signals: .....".
- Time-Switching Protocol section - Changed all occurrances of "Fccu_eout[1]' to EOUT[1].
- Time-Switching Protocol section - Added foot note: Foot note reads "Actual value depends on the design settings at pad level."
- Bi-Stable Protocol section - Changed all occurrances of "Fccu_eout[1]' to EOUT[1].
- Bi-Stable Protocol section - Added foot note: Foot note reads "Actual value depends on the design settings at pad level."

- Changed "Error" to "Fault throughout chapter.
- Acronyms and Abbreviations section -- removed conditional text tag for "non-critical Fault" in NCF row.
- Bi-stable protocol section -- Added footnote to "High-Z" entry of reset row of Bi-stable encoding table. Also added same footnote in text. "Actual value depends on device specific settings at pad level."
- Dual Rail protocol section -- Added footnote to "High-Z" entry of reset row of Bi-stable encoding table. Also added same footnote in text. "Actual value depends on device specific settings at pad level."
- Time Seitching protocol section -- Added footnote to "High-Z" entry of reset row of Bi-stable encoding table. Also added same footnote in text. "Actual value depends on device specific settings at pad level."
- FOSU section - Made minor non-technical XML coding corrections.
- Fault priority scheme and nesting section: Added text: If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

- FCCU FOSU Output Supervision Unit -- Added following note: -- FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or destructive resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

- Editorial changes.
- Added the note "The reset value is chip-specific. See the Chip Configuration chapter for details." at the beginning of the following sections:
  - RF Configuration Register (FCCU_RF_CFG*n*)
  - RFS Configuration Register (FCCU_RFS_CFG*n*)
  - RF Enable Register (FCCU_RF_E*n*)
  - RF Time-out Enable Register (FCCU_RF_TOE*n*)

- Introduction section -- removed text referring to FCCU circuitry being checked at start up.
- EOUT interface section: removed reference to FCCU failure inputs table.
- Use cases and limitations section -- Added note to bullet item "Software reset is optional." stating that "This reset is not implemented on all devices".

- In Error signal flow, deleted the signal-flow figure and replaced it with a reference to the chip-specific FCCU information (where the figure now exists).

- Editorial corrections

- EOUT interface
  - Reorganized the sentence about resets' effect on failure indication pins as two sentences and a list
  - In the list, added "FOSU time-out reset" as a reset type that influences the state of failure indication pins

- Changed reset value of FCCU_CTRL register from 0x0000_0000 to 0x0000_00C0. (OPS field changed from 00b to 11b.)

- In NCF Fake Register (FCCU_NCFF), changed register description.

- INTRODUCTION SECTION - Change "Internal reactions (independently configurable for each NCF):" to "Configurable internal reactions for each NCF:"
- INTRODUCTION SECTION - Change "External reaction (failure is reported to the outside world via one or more output pins)" to "External reactions via configurable output pins"
- INTRODUCTION SECTION - Change bullet item text to "Configurable fault control" (was "Configurable and graded fault control").
- INTRODUCTION SECTION - Removed phrase "for application/test/debugging purposes"
- INTRODUCTION SECTION - Changed "FCCU is designed for assuming that the system clock is faster than the IRC clock" to "The FCCU is designed to function when the system clock is faster than the IRC clock."

• FSM description - Change "After reset lifts, the state shall be transiently locked (permanent lock ? is unlocked and transient lock ? is locked)." to "After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked." • In the FCCU_CTRLK register description, change note "There should not be any other operation in between the above steps." to "The FCCU_CTRLK and FCCU_CTRL registers must be written with consecutive instructions. Both registers must be written as 32-bit values. Do not use read-modify-write instructions, such as bit field instructions, to modify these registers." • In section FCCU_DELTA_T, FCCU_IRQ_ALARM_ENn, FCCU_NMI_ENn, FCCU_EOUT_SIG_ENn descriptions: Changed, "This register can be written only when the FCCU is in CONFIG state." to Note. • In FCCU_CFG_TO description changed "oscillator" to "IRCOSC". Changed part of description to Note. • In section Self-checking capabilities, removed "Self-checking reaction (reset)" figure.
• For FCCU_NCF_CFGx, FCCU_NCFS_CFGx, FCCU_NCFEx, FCCU_NCF_TOEx and MCS registers: Changed the reset value of the registers from 0x0000_0000 to X. Added the following footnote to each register: The reset value is chip-specific. See the chip-specific FCCU information. Added the following note to register descriptions: The reset value is chip-specific. See the chip-specific FCCU information. For all configuration registers, made the following text into a note: These registers can be written only when the FCCU is in CONFIG state. • INTRODUCTION -- Changed bullet item "Configurable and graded fault control" to "Configurable fault control" • INTRODUCTION -- In bullet item "Permanently locked until next reset ...., Changed "FCCU_TRANS_LOCK" to "FCCU_PERMNT_LOCK" • INTRODUCTION -- Changed "failure indication" to "error out" • MAIN features -- Changed "Failure indication" to "Error indication" • FSM description -- Changed reference to state diagram. • FSM description -- Changed "Following one of the following events" to "Following state transions occur on one of the following event" • Corrected notation for binary numbers. • FCCU state diagram -- Changed block text on CONFIG state from "AND NOT (configuration locked)" to "AND (configuration unlocked)" • FCCU state diagram -- Added block text to ALARM state -- "Actual text in diagram is "Any Fault Pending AND FCCU_IRQ_ALARM_ENn"" • EOUT interface -- Throughout section changed uppercase "ERROR" to "FAULT". • Deleted the text -- "The EOUT frequency is generated by dividing IRCOSC clock by a fixed factor of 2^18."
• At the end of EOUT interface section, added NOTE on state of EOUT pads.
• FCCU_MCS register -- added note: The reset value is chip-specific. See the chip-specific FCCU information. • FCCU_CTRL register, FILTER_WIDTH field -- Changed 10 bit description from "filters glitches up to 100 microseconds for 20 Mz IRC clock" to "filters glitches up to 100 microseconds". Changed 11 bit description from "filters glitches up to 100 microseconds for 20 Mz IRC clock" to "filters glitches up to 100 microseconds"
• Added the following note to the FCCU_NCFS_CFGn and FCCU_NMI_ENn register descriptions: Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time. • The following text was added to the EIN1 bit description of the FCCU_EINOUT register -- While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[1] pin. The following text was added to the EIN0 bit description of the FCCU_EINOUT register -- While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[0] pin.
• Acronyms and Abbreviations section: Added IRCOSC entry into the Acronyms and Abbreviations table. • FCCU_CTRL register, removed "configure the glitch filter present on location given by FILTER_POSITION parameter" • FCCU_CTRL[DEBUG] description, changed ipg_debug to debug. • Added the note "The reset value is chip-specific. See the Chip Configuration chapter for details." at the beginning of the following sections: NCF Configuration Register (FCCU_NCF_CFGn), NCFS Configuration Register (FCCU_NCFS_CFGn), NCF Enable Register (FCCU_NCF_En) and NCF Time-out Enable Register (FCCU_NCF_TOEn)
• Minor non-technical grammatical corrections. • FCCU_CFG_TO register description - replaced "See FCCU register set table." with "The description for a configuration register will contain a NOTE that it is writable only in the CONFIG state.”
• Dual Rail Protocol section -- Corrected error phase portion of timing diagram. • FCCU_STAT register, STATUS field -- Changed "UNKNOWN" to "Reserved" in bit field descriptions. • Use Cases and Limitations section, Misconfigurations heading -- removed "Software reset is optional. NOTE: Software reset is not required for all devices."

• In Definitions, after the list of applicable reset types, added this information: *For more information on each type of reset, see the MC_RGM and reset chapters.*

A.87 STCU2 module changes

• Corrected reset value of STCU2_CRCE register from 0x0000_0000 to 0xFFFF_FFFF.
• Block Diagram -- changed "errors" to "faults" and "safety" to "reliability" • Main Features section -- changed "errors" to "faults" and "safety" to "reliability" • Write Access Time Out section -- Changed "safety feature" to "malicious access protection feature" • On Line Self Test Sequence -- Removed "In case the L/MBIE have been enabled, an interrupt is generated." • Changed bit 0 field description for the LBRMSWx bits of the LBRMSW register to "Reserved"
• Added the following note to the IPS bus interface section: "When writing to the online registers, the user must wait for the expiration of the offline key. The time the user should wait is (IPS_CLK)*4096 clock periods. If the IPS_CLK is 66.5 Mhz then the user must wait (1/66.5)*4096 = 61.6 us. • Added footnote to CRCE register for default reset value. "See the chip specific information for the default reset value of this field."

• Added the note "The reset value is chip-specific; see the chapter that describes how modules are configured and connected." at the beginning of the following sections:
  • STCU2 Configuration Register (STCU2_CFG)
  • STCU2 PLL Configuration Register (STCU2_PLL_CFG)
  • STCU2 Watchdog Register Granularity (STCU2_WDG)
  • STCU2 CRC Read Status Register (STCU2_CRCR)
  • STCU2 Error FM Register (STCU2_ERR_FM)
  • STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)
  • STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)
  • STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)
  • STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)
  • STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)
  • STCU2 LBIST Control Register (STCU2_LB_CTRL*n*)
  • STCU2 LBIST PC Stop Register (STCU2_LB_PCS*n*)
  • STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL*n*)
  • STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH*n*)
  • STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL*n*)
  • STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH*n*)
  • STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL*n*)
  • STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH*n*)
  • STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW*n*)
  • STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW*n*)
  • STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW*n*)
  • STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW*n*)
  • STCU2 MBIST Control Register (STCU2_MB_CTRL*n*)
• In STCU2 CRC Expected Status Register (STCU2_CRCE), added the note "See the chip specific information for the default reset value of this field." at the beginning of the section.

• Hardware ABORT management section -- Rewrote this section for grammatical clarity.
• In the block diagram, indicated that the CRC block is optional.
• Minor non-technical typographical corrections.
• In Offline self-test sequence, changed *PMOSEN or CHKBRD* to *PMOSEN or MBU*. • In Online self-test sequence, changed *PMOSEN or CHKBRD* to *PMOSEN or MBU*.
• In the value descriptions for the SHS field in the STCU2_LB_CTRL registers, changed *(bist_clk)* to *STCU2 core clock*.
• Specified the reset value of the CRCE register to be undefined at the release of reset.
• In the description for the PTR field in the STCU2_LB_CTRL registers, added this note: *The value of NLBIST is device specific. See the device specific information for the value of NLBIST*

• In the STCU2_LBRMSW register description removed reference to "specific functional reset".
• Added the following note to the Built-in self-test scheduling section: Program operation should ensure that no flash write/erase operations are ongoing when online LBIST is triggered. Otherwise, a flash segment may be corrupted.
• Added the following note to the CRC section: "The CRC calculation depends on the internal state of the STCU2 when the online/offline self test is started. The STCU2's internal state is dependent on DCF values and their programming sequence for offline selftest. The calculated CRC value can be determined by running an LBIST test under known conditions. The CRC unit produces a stable and predictable result but is dependent on several stimuli (including programming order and the initial internal state of the STCU2 design) during online self test. To calculate a CRC value, run an LBIST test on known good silicon with a predefined software programming sequence. Record the CRC result value. Then provide the STCU2 programming sequence and the initial register vlaues for further runs if the CRC feature is to be used. Once a known good CRC value is determined, the CRC value will not change from one run to another so long as the user does not change the STCU2 programming sequence or any of the values used to program STCU2 registers. To obtain a consistant CRC calculated value, the STCU must start from the same state compared to the state when the CRC was originally evaluated."
• Added the following to the STCU2_LB_CTRL[CSM] field description: "The next LBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it is scheduled sequentially to the completion of the current LBIST execution."
• Removed the following from the STCU2_LB_CTRL[PTR] field description: "The next LBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it is scheduled sequentially to the completion of the current LBIST execution." • Reformatted the STCU2_LB_CTRL[PTR] field description.
• In Register write-access watchdog timer, added the following note: *Each DCF record takes 8 STCU2 clock cycles for write completion. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is Value divided by 8. To refresh the hardware Watchdog, the user must write key 2 into the STCU2_SKC register.* • Changed the access type of the STCU2_MBUFML, STC2U2_MBUFMM and STCU2_MBUFMH registers from read-only (RO) to read/write (RW).
• In the register descriptions for STCU2_LB_CTRL, STCU2_LB_PCS, STCU2_LB_PRPGL, STCU2_LB_PRPGH, STCU2_LB_MISREL, STCU2_LB_MISREH, STCU2_LB_MISRRL, STCU2_LB_MISRRH, STCU2_LB_MISRELSW, STCU2_LB_MISREHSW, STCU2_LB_MISRRLSW, and STCU2_LB_MISRRHSW, deleted this text:*The offset of this register is a function of NLBIST.* • Revised the description for the PTR field in the STCU2_LB_CTRL registers.
• In Online self-test sequence, added the following note: *Flash memory must be in the idle state when BIST execution is started. Program and erase functionality must not be enabled when entering BIST execution.* • In Design implementation information, added the following caution notice: *Do not execute a BIST on more than one partition at a time.*
• In STCU2_CFG register description, added description and table describing three MBIST test types and their coverage. • Modified STCU2_CFG[PMOSEN] and STCU2_CFG[MBU] field descriptions and bit definitions.

• Retitled the Built-in self-test scheduling section.
• In Watchdogs, retitled the following sections:
  • Initialization watchdog timer
  • Built-in self-test watchdog timer
  • Register write-access watchdog timer

• Edited Design implementation information for grammar and style.
• In Design implementation information, changed *Do not execute a BIST* to *Do not execute LBIST*.
• In Design implementation information, corrected the grammar in the caution notice.
• In IPS bus interface, replaced the note with a reference to the chip-specific STCU2 information (which is where the correct contents now appear).

A.88 REG_PROT module changes

- In Memory map and register definition, made editorial changes.

A.89 PASS module changes

- Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*), LOWLOCK and MIDLOCK fields - Removed footnote: Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
- Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*), ATSL reset value - Added "If no DCF records have been written to change the ATSL bit lock state, the bit will reset to ‘1’.
- Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*), HIGHLOCK field - Removed footnote: Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
- Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*), L_256LCK field - Removed footnote: Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
- Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*) - Revised the reset-value footnotes.
- Activation Section -- Changed "Jitter Activation constant in BAF Flash should be programmed to make the most significant bit ‘0’. to "Jitter Activation constant in BAF Flash should be programmed to make the least significant bit ‘0’.
- Clock Jitter Truth table -- Corrected bit pattern in Clock Jitter Enable Column to reflect that CJE bit is the LSB.
- Clock Jitter Truth table -- Added "OEM Production" to first row in Lifecycle column.

- In the "Memory map and register definition" section, changed the note "Non-implemented registers generate bus aborts if enabled by the SOC" to "Non-implemented registers generate bus aborts if enabled".

- In Life Cycle Status Register (PASS_LCSTAT), in the CNS field description, deleted the text "See section Censoring and uncensoring the device."

- In Setting lock bits in a password group in Table 88-6 changed the column heading from "DCF CS/ADDR" to "DCF Command Word"

- Changed Reserved bit of LCSTAT register to read-only and always 1
- Added following text to description of JUN bit in LCSTAT register: "(In life cycle FA, the JUN bit will always be 0.)"
- In Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*) description, added sentence: "Flash Utest region controlled by LOCK3[0] is read unlocked in Failure Analysis life cycle."
- Incorporated minor, non-technical editorial and formatting corrections.

- In Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*) register description, moved paragraph: "LOCK1_PGn register functions are as shown in this section."
- In Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*) register, RL0, RL1, RL2 and RL3 field footnotes modified.

- Challenge Selector Register (PASS_CHSEL) description -- Changed “When a master accesses this register, the PASS module stores the number of the master in the LOCK3_PGn register of the selected passgroup and sets the PGL bit in that register.” to “When the Master Only (MO) bit of the PASS_LOCK3_PGn register is set, only the master that unlocked access to the passgroup settings will be able to access these registers. The CMST field of the Challenge Status Register will reflect the locking master. When the Master Only (MO) bit of the PASS_LOCK3_PGn register is not set, the passgroup settings can be accessed by any master.”
- Challenge Selector Register (PASS_CHSEL) description -- Changed “ The ID of the accessing master sets the CMST field of the Challenge Status register.” to “When a master writes this register, its ID sets the CMST field of the Challenge Status register.”
- Challenge Input Register (PASS_CIN*n*) description -- Added following text to second paragraph: “In case of a successful password comparison, the PASS module stores the ID of the master which has written CIN0 to CIN7 in the PASS_LOCK3_PGn register of the selected password group, as well as sets the PGL bit of that register.”
- In Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*) description -- changed NOTE to CAUTION
- Revised Table 88-7.
- Incorporated minor, non-technical editorial and formatting corrections

- PASS_LOCK1_PGn register description -- Minor formatting and grammatical changes.

- Added the following note to the Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*) description: note - Do not toggle DBL and RLx bits with the same write command. Use one instruction to program the DBL bit and a separate instruction to toggle the RLx bits.

- Debug Interface Access :
  - Deleted "The HSM can further block Debug Interface Access, even if it enabled by PASS. This allows HSM to granted Debug Interface Access after an encrypted secure challenge/response protocol with an external tool -- see HSM documentation".

- In Challenge Selector Register (PASS_CHSEL), changed GRP field from 2 bits to 3 bits in order to accommodate eight Password Groups.

- Revised Challenge Input Register (PASS_CIN*n*) description to indicate that a successful password comparison "clears the PGL bit of that register" (was "sets the PGL bit...").

A.90 TDM module changes

- Editorial changes.
- Overview :
  - Correction: Added "Software tamper region override" to list of security data passed to the TDM by the SSCM module.
  - Clarification: Reworded note regarding PASS module settings having priority over TDM settings.
- Flash erase counter and tamper detection :
  - Clarification: For each TDR, software must ensure that the tamper diary is not full before writing a diary record. If it is full the TDR can either be overridden or, if an unused TDR is available, a duplicate TDR can be defined.
  - Following extraneous sentence deleted: "A schematic representation of the TDM implementation is depicted in the following figure."
- Tamper Region Override (TO) DCF client :
  - Changed the description for TOE[5:0] in the table: A '1' in the TOE field enables the blocks assigned to a tamper region to be erased without creating a diary record. Previously states that a '1' would allow the diary to be erased.

- In TDR Status Register (TDM_TDRSR), added the note "The reset value for this register is device specific. See the chip configuration chapter." at the beginning of the section.

- Editorial changes.
- Overview :
  - Clarification: The Tamper Detection Module provides a type of flash memory *erase* protection mechanism. (previously stated *write* protection mechanism).
  - Added list of three dedicated signals provided by TDM module.
- Features :
  - Deleted from feature list: "Memory mapped registers interface for reading via software of specific registers."
- DCF clients :
  - Deleted: All content describing the general function of DCF records.
- TDR Status Register (TDM_TDRSR) :
  - Register description rewritten for clarification.
  - Statement deleted: "This register is not defined in all devices. See the chip configuration chapter for details."
- Last Flash Programmed Address Register (TDM_LFPAR) :
  - Statement deleted: "This register is not defined in all devices. See the chip configuration chapter for details."
- Diary Base Address (TDM_DBA) :
  - Footnote added to register: "The reset value of this register is set via DCF record."
- Software Tamper Override Key Region (STO_KEYn) :
  - Correction: This register is write-once. Previously stated as being write once per reset cycle.
  - Clarification: The DCF client associated with this register is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.
- Flash erase counter and tamper detection :
  - Correction: After TDM verifies that a successful double word program to a valid TDR diary location, TDM ensures the blocks assigned to a TDR are unlocked for erase until a successful erase operation is performed.

(previously stated the blocks are unlocked until an erase is performed *with program completion interrupt enabled or till next hard reset*.
• Specifying the diary base address :
  • Clarification: Statement that diary can hold "a maximum of 256 x 8 byte entries" changed to "...each diary region limits the erases of the assigned blocks to 256 erases.
• Diary :
  • Correction: The diary base address DCF client is only writable once (previously stated it is once during reset).
• TDR lock status. :
  • Section title changed (was "Diary Base Address Verification").
• TDR override :
  • Section title changed (was "Diary management").

• Software Tamper Override Key Region (STO_KEYn) :   • Changed access from R(ead only) to RW.
• Updated Figure 89-1. • Removed all references to "C55FMC".
• In DCF clients in Table 89-1 column for Strategy, added "write once" for: • Software Tamper Region Override Disable. • In every subsection of DCF clients changed "Warning" notice to "Caution" notice. Changed introductory descriptions in all One Time Programmable Enable (OTPENn) DCF client sections. • For Software Tamper Override Key Region (TDM_STO_KEY*n*) ensured the correct register array size.
In Specifying the diary base address, in the sentence, "The DCF client is writable once during reset and the default bit state is ‘0’." deleted "during reset" and change default bit state to '1'.

• Editorial changes.
• Changed second Caution bullet in:
  • Diary Base Address (DBA) DCF client.
  • Software Tamper Region Override Disable (STO_DIS) DCF client.
• Edited note re. TDR Status Register (TDM_TDRSR) reset value.

• Overview :
  • Deleted list of three dedicated signals provided by TDM module.
• Features :
  • Added feature list item: "Memory mapped registers interface for reading via software of specific registers."
• Table 89-2 : Added note stating boundary requirement for TDM diary base address.

Table 89-1 : Changed table title to "DCF clients" (was "DCF client address map").

A.91 Wakeup Unit changes

• No substantial content changes

MPC5777M Emulation and Debug Device Reference Manual

Document Number: MPC5777MBDRM
Rev. 3 11/2013

Contents

Section number **Title** **Page**

Chapter 1
Emulation and Debug Device Introduction

Chapter 2
Buddy Device—Crossbar and RAM (BD_XBS_RAM)

Chapter 3
Buddy Device—System Integration Unit Lite (BD_SIUL2)

Chapter 4
Buddy Device—Debug and Calibration Interface (DCI)

Chapter 5
Buddy Device—JTAG Controller (JTAGC)

Chapter 6
Buddy Device—JTAG Master (JTAGM)

Chapter 7
Buddy Device—Nexus Aurora Router (NAR)

Chapter 8
Buddy Device—Nexus Aurora Link (NAL)

Chapter 9
Buddy Device—Nexus Aurora PHY (NAP)

Chapter 10
Buddy Device—Data Write Processing Unit (DWPU)

Chapter 11
Buddy Device—Nexus Read Write Access (RWA)

Chapter 12
Buddy Device – LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug

Chapter 1
Emulation and Debug Device Introduction

1.1 Introduction

The emulation and debug device is a multi-die MCU that adds additional functionality specifically for ease in calibration and debug. The Emulation and debug device (ED) consists of the production die (PD) and a buddy die (BD).

1.2 Functionality

The ED is a PD packaged with a BD to provide additional capability during development. The ED is used for development, debug, and calibration of automotive powertrain controller applications for 4- to 8-cylinder engines. To enable these operations, the buddy device supports the following functions:

- Debug and calibration tool interface—Allows tool read/write access to buddy device resources even if main MCU supplies are removed as long as special ED debug supplies are maintained. The BD duplicates the LFAST, Debug and Calibration Interface (DCI) and JTAG Master Module (JTAGM) functionality that is also available in the PD.
- High speed tool trace port (based on the IEEE-ISTO 5001-2012 Nexus Aurora standard)
- 2 MB on-chip SRAM usable as calibration overlay or trace RAM—entire SRAM contents are maintained as long as ED debug supply is maintained

1.3 Feature summary

On-chip modules available on the BD include the following features:

- 2 MB on-chip SRAM
  - Access timing matches internal PD flash accesses when used in conjunction with PD calibration remap logic
  - Includes ECC
  - Implemented as 4 × 512 KB partitions where each partition can be accessed by different masters without causing blockages to access of any other partition
- XBAR switch architecture
- XBAR slave interface
  - 64-bit data width with ECC checkbits
  - Supports "Early Write" AHB protocol (AHB-EW) for one clock trace data writes
  - Allows a PD with AHB master to support read and write to buddy device memory-mapped resources—Memory-mapped resources include buddy device SRAM and AIPS resources
- Nexus Aurora auxiliary trace port
  - 1.25 Gbit/s maximum data rate per lane
  - Four lanes supported in ED package
  - Port configurable to operate in 2- or 4-lane width
  - One lane LVDS reference clock in
  - Supports Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2012 standard.
- Data Write Processing Unit (DWPU)
  - Filters data trace stream to retain only writes to user selected locations
  - Detects user configured out of range condition on writes to selected locations
- Debug and Calibration Interface—Supports JTAG (IEEE 1149.1 and IEEE 1149.7) mode and LFAST mode
- Independent high and low voltage supplies allow tool control of BD debug and calibration resources even when the main MCU supplies are off
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)

1.4 Block diagram

This figure shows the top-level block diagram of the emulation and debug device.

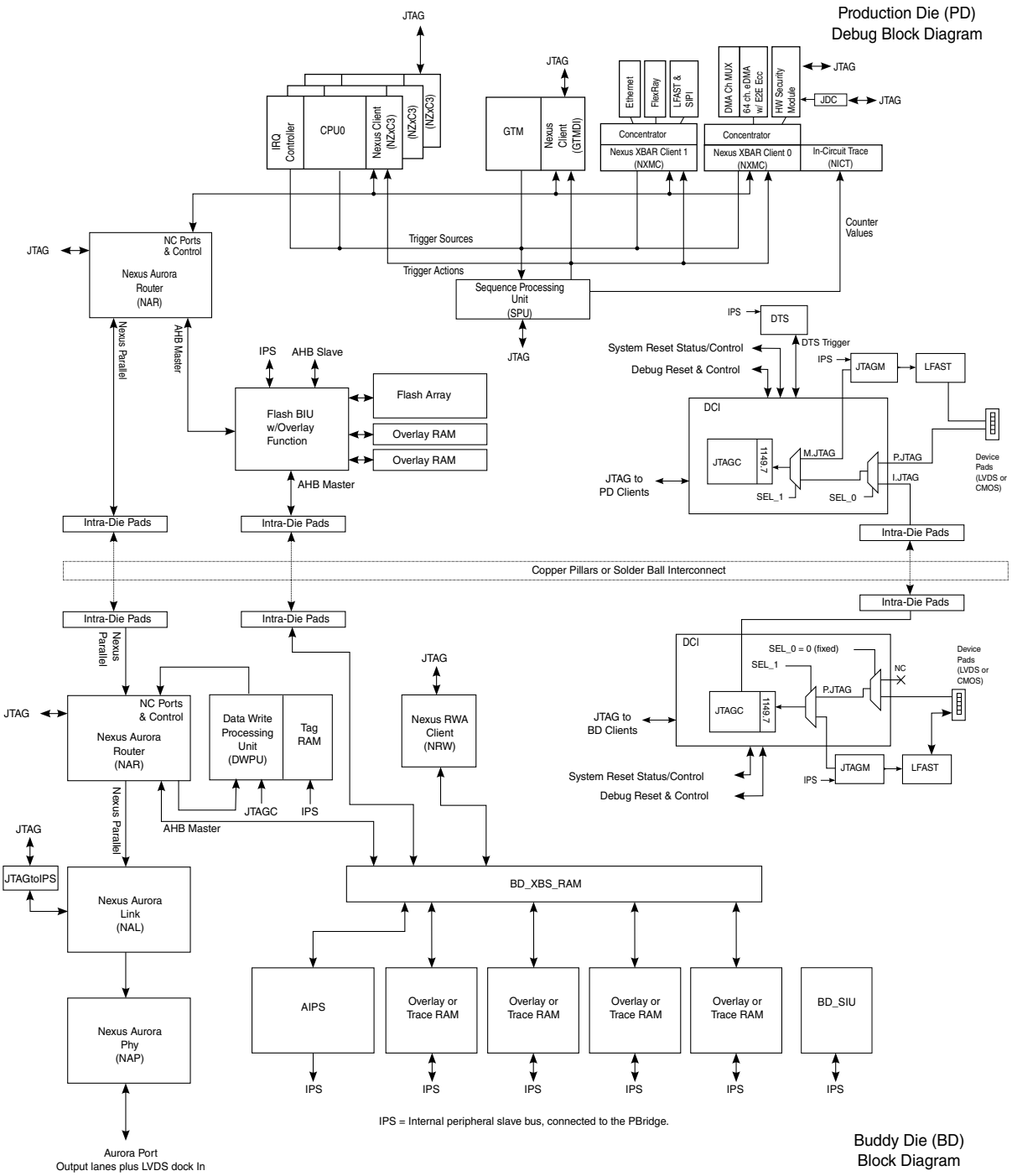


Figure 1-1. Emulation and debug device architecture

1.5 Clocking

The figure below shows the clocking structure of the BD. There is one internal clock source and four external clock sources:

- Internal RC oscillator (IRC) clock
- System clock coming from the PD (identified as BD_CLK clock in the clock generation figure of the Clocking chapter)
- LFAST PLL reference clock coming from the PD (BD LFAST PLL clock in the clock generation figure of the Clocking chapter)
- JTAG clock coming from the TCK pin in the ED
- Nexus Aurora PHY clock coming from differential LVDS pins in the ED

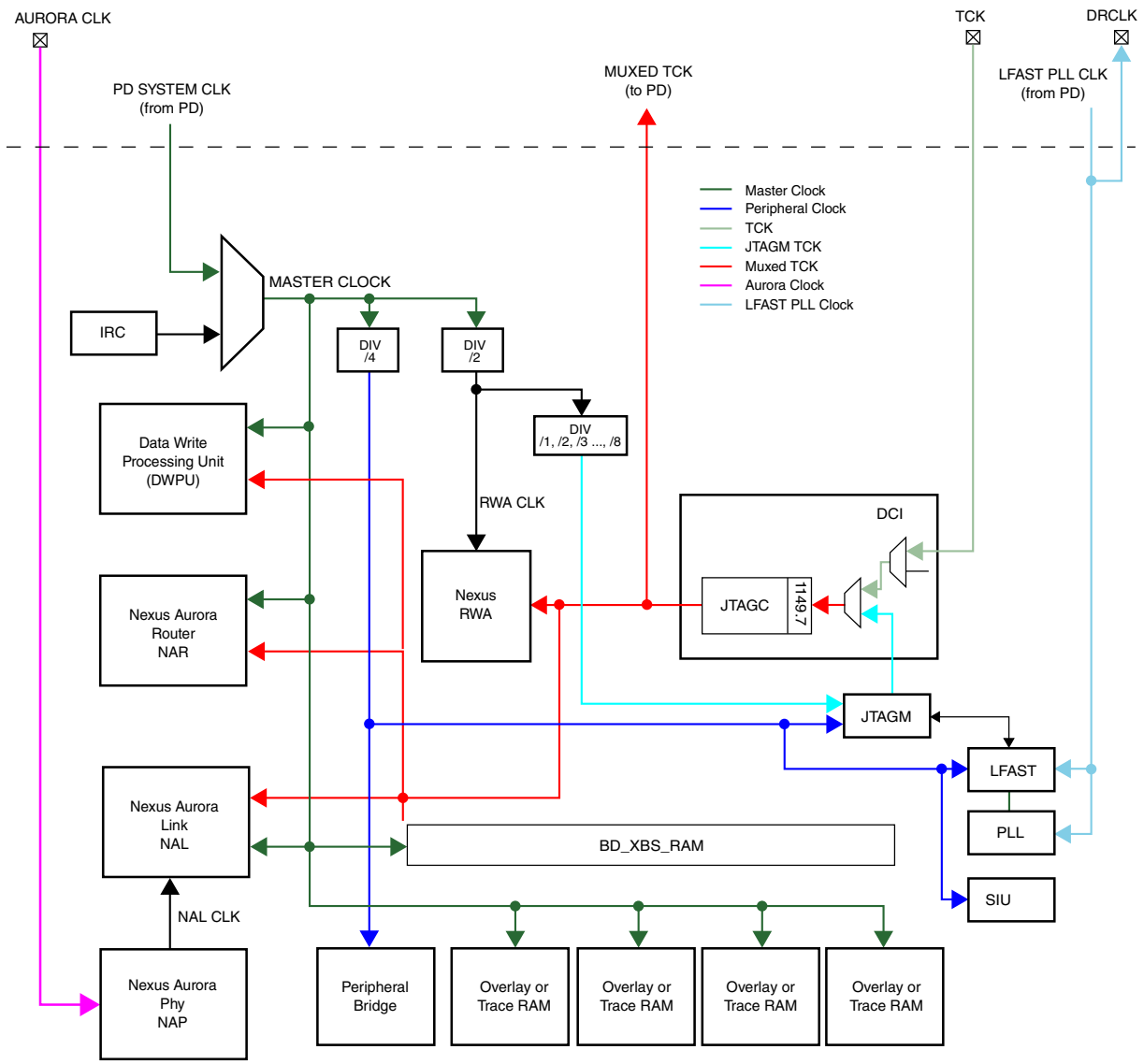


Figure 1-2. BD clock structure

The IRC is the source of the BD master clock when the PD is unpowered, or powered but still in reset state. It has a nominal frequency of 64 MHz, does not have any user accessible register and is always enabled. It does not have any trimming capability, therefore the expected precision is on the order of ±30%.

The PD system clock is the source of the BD master clock when the PD is powered and out of reset state. When the PD PLL is engaged and locked this clock has a nominal frequency of 200 MHz, but while the PLL is not engaged the frequency is that of the PD internal RC oscillator.

The master clock directly feeds the XBAR, SRAMs and the Aurora sub-system. It is also divided down to generate the following internal clocks:

- Peripheral clock (divided by 4)—provides the clock to memory mapped registers and other low-speed logic
- Nexus RWA clock (divided by 2)—provides the clock to the Nexus Read Write Access module
- JTAGM TCK (divided by 1, 2, …, 8)—provides a high speed TCK clock to the JTAGM module

The LFAST PLL clock from the PD (also referred to as lfast_sysclk) feeds the LFAST PLL, but it can also directly clock the LFAST in the low data rate mode. This clock is also made available as an output clock to the DRCLK package pin.

The internal TCK clock (referred to as muxed clock in Figure 1-2) is used in the Nexus RWA, NAR, NAL, and it also goes to the PD to be used as the JTAG clock. (In the ED, the JTAG package pins are bonded to the BD JTAG interface; therefore, the PD receives the JTAG signals from the BD.) This clock can come from two sources (see the Buddy Device–Debug and Calibration Interface (DCI) chapter):

- TCK package pin, when using IEEE 1149.1 or IEEE 1149.7 access
- Divided down version of the master clock to be used in LFAST/JTAGM debug accesses

Note

The division factor for the LFAST/JTAGM clock is defined by the TCKSEL field of the JTAGM_MCR (refer to the JTAGM chapter for more details). However the division factor described there applies to an already divided-by-2 clock, as shown in Figure 1-2.

When the PD is unpowered, access through LFAST/JTAGM is not possible. In this case, JTAG accesses can be done only by IEEE 1149.1 or IEEE 1149.7 modes accessing through the Nexus RWA module. In this mode, the TCK clock comes from the package pin and the RWA clock is a divided-by-2 version of the IRC; therefore, these clocks are asynchronous to each other. When clocks at the input of the RWA are asynchronous, there is a restriction that the RWA clock must be at least double the frequency of the JTAG clock. This means that the master clock frequency must be at least four times the JTAG clock frequency.

In the LFAST/JTAGM mode of operation, both RWA clock and internal TCK are synchronous to each other, therefore they can both be the same frequency. This means that TCK frequency can be half of the master clock frequency. This table summarizes the frequency restrictions on the Nexus RWA accesses.

Table 1-1. JTAG operating frequencies

PD configuration	BD configuration	PD system clock frequency [MHz]	BD master clock frequency [MHz]	Max RWA JTAG frequency [MHz]	Notes
Unpowered or in reset state	1149.1 or 1149.7	—	44.8	11.2	Assuming 64 MHz BD IRC with 30% tolerance
Powered, out of reset, PLL not engaged, system clock at IRC frequency	1149.1 or 1149.7	15.52	15.52	3.88	Assuming 16 MHz PD IRC with 3% tolerance
Powered, out of reset, PLL engaged	1149.1 or 1149.7	200	200	27.7/50	JTAG speed is limited by TCK negedge to TDO delay (14 ns) plus external setup to TCK posedge (4 ns). Otherwise, the maximum Nexus RWA JTAG frequency is internally limited to 50 MHz.
Powered, out of reset, PLL engaged	LFAST/JTAGM	200	200	100	—

1.6 Reset logic

There are five sources of reset that are combined to generate the reset of system portions as described in the following table:

1. BD power-on reset—asserts immediately upon assertion of BD POR, and remains asserted for 10 clock cycles after negation of POR.
2. PD reset—asserts two clock rising edges after assertion of reset signal coming from PD and remains asserted while the PD reset signal is asserted plus two clock rising edges after the PD reset signal is de-asserted (this reset source is reflected in the PD_RST bit of the BD_SIU_RSR register).
3. DCI FBD reset—asserts immediately when the FBD_RESET bit of the DCI Control Register is set, and remains asserted for one clock cycle after FBD_RESET is negated.

4. JTAG controller reset—asserts immediately when a reset request signal is received from the JTAG Controller by the execution of EXTEST, CLAMP, HIZ boundary instructions, and remains asserted for one clock cycle after the reset request is negated.
5. Nexus reset—asserts by DCI logic based on JCOMP state and JTAG source selected. DCI FBD also generates a Nexus reset.

In the following table, an 'X' indicates that the reset source generates a reset signal to the specified block. JTAG mode only indicates that those blocks are reset by the corresponding reset sources only in the case where the debug interface is running through the JTAG external pins. When the debug interface is running through the LFAST interface or by software through JTAGM, those blocks are not reset by the mentioned reset sources.

Table 1-2. Reset sources and functional blocks

Blocks	BD POR	PD reset	DCI FBD reset	JTAGC reset	Nexus reset
DCI	X	—	—	—	—
Z0 Nexus RWA	X	—	X	—	X
BD XBAR	X	—	X	X	—
Overlay/Trace RAM	X	—	X	X	—
JTAGM	X	—	JTAG mode only	JTAG mode only	—
LFAST	X	—	JTAG mode only	JTAG mode only	—
NAR/DWPU	X	—	X	X	X
NAL	X	—	X	X	X
NAP	X	—	X	X	X
BD SIU	X	—	X	X	—
BD SIU GPIO registers	X	X	—	X	—
BD INIT flag	X	—	X	X	—
PD/BD interface logic	X	—	X	X	—

NOTE

The PD reset is not affected by the BD reset. The BD does not interfere with startup or shut down timing of the PD.

1.7 BD memory map

The BD memory map is contained within a 16 MB range. This entire range is accessed through the PD Flash BIU. The BD memory map is divided into two regions: overlay RAM and device registers.

The device register region allows the CPUs on the PD to access configuration registers of the peripherals in the BD, such as LFAST, JTAGM, XBAR configuration, etc. The device register region are divided into 16 KB regions for assignment to each peripheral block.

The memory map for the BD is shown below as an absolute address relative to the PD address map. Within the BD, only a 24-bit address bus is required. The upper 8 bits of the complete 32-bit address are decoded and removed by the PD PFLASH controller.

Table 1-3. BD memory map

Start address	End address	Allocated size (KB)	Description
BD overlay RAM			
0x0C00_0000	0x0C0F_FFFF	1024	Extended overlay RAM
0x0C10_0000	0x0C1F_FFFF	1024	Extended BD overlay RAM
0x0C20_0000	0x0C3F_FFFF	2048	Reserved BD overlay RAM
0x0C40_0000	0x0C5F_FFFF	2048	Reserved BD overlay RAM
0x0C60_0000	0x0C7F_FFFF	2048	Reserved BD overlay RAM
BD registers			
0x0C80_0000	0x0C80_3FFF	16	BD_SIUL2
0x0C80_4000	0x0C80_7FFF	16	LFAST
0x0C80_8000	0x0C80_BFFF	16	JTAGM
0x0C80_C000	0x0C80_FFFF	16	DWPU
0x0C81_0000	0x0C81_3FFF	16	Reserved
0x0C81_4000	0x0C81_7FFF	16	Reserved
0x0C81_8000	0x0C81_BFFF	16	Reserved
0x0C81_C000	0x0C81_FFFF	16	Reserved
0x0C82_0000	0x0CFF_FFFF	8064	Reserved

Chapter 2
Buddy Device—Crossbar and RAM (BD_XBS_RAM)

2.1 Introduction

Within the buddy device (BD), there is a crossbar switch that provides the required connections between multiple bus masters and the overlay RAM. For a diagram of the Emulation device (ED) debug architecture that shows the BD blocks and crossbar, see the Emulation device introduction chapter. This section provides details on the structure of this function.

Note

Throughout this description, the term "overlay RAM" is applied to the BD memory, even though this memory can be used for *both* overlay and debug trace information.

The 64-bit BD crossbar and RAM structure consists of:

- Input registers to capture transfers (address, attributes and write data) targeted at the overlay RAM or memory-mapped peripheral space
- A 3 x 5 crossbar switch to route requests from masters to either the overlay RAM or the system bus
    - Bus masters include the BD's Nexus Aurora Router (BD-NAR), the production device's flash controller (PFLASH) and the BD's Nexus Read/Write Access Client (RWA)
    - Bus slaves include 4 equal-sized regions of the overlay RAM and a peripheral bridge (AIPS-Lite) bus controller
        - 160-bit RAM arrays (128 bits of data, 32 bits of end-to-end ECC)
        - End-to-End ECC32 address and data protection

- ECC scheme based on 29-bit address and 32-bit data
- Overlay RAM supports a 64-bit "early write" datapath and a 128-bit read datapath

- Output alignment logic and registers to capture read data being driven to the requesting master

The result is an efficient structure supporting concurrent references of the overlay RAM for calibration accesses from the PFLASH controller on the Production Device (PD) and debug references from the other masters. The crossbar and RAM support 200 MHz operation.

Note

Calibration reads from the PFLASH have the same timing characteristics as normal data reads from the flash array.

2.2 3 x 5 Crossbar switch and attached controllers

In the same manner as the comparable crossbar switches in the PD, the BD crossbar concurrently supports up to 3 simultaneous connections between the master and the slave ports. It implements a 64-bit "early write" datapath and a 128-bit read datapath and fully supports concurrent accesses between calibration reads and debug trace writes to different overlay RAM regions.

This crossbar implements fixed priority arbitration on a per slave port basis where the master number defines the relative priority in a reverse numeric order, that is, m0 > m1 > m2 (PFLASH > BD-NAR > Nexus RWA). Each slave port is "parked" on the last master making a request.

The design supports {8, 16, 32, 64}-bit SINGLE reads, {8, 16, 32, 64}-bit SINGLE writes along with 64-bit WRAP4 burst reads and writes. Two consecutive sequential 128-bit accesses of the RAM are used (only) on the WRAP4 burst read.

2.3 BD_XBS_RAM End-to-End ECC32 Checkbit/Syndrome Coding Scheme

The e2eECC scheme used in the BD overlay RAM is a simplified derivative of the algorithm used in the Production Device. The only difference is the code is based on a 32-bit data field, but the same H-matrix is used with the appropriate 32-bit data upper or lower word zeroed.

Consider the following Boolean equation definitions:

e2eECC64[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0] ^ ecc_data_lower[7:0]

where `ecc_addr` is H-matrix evaluation of the 29-bit access address (addr[31:3]), `ecc_data_upper` is the H-matrix evaluation of the upper data word (data[63:32]) and `ecc_data_lower` is the comparable evaluation of the lower data word (data[31:0]).

For the BD RAM definition, the following (re)definitions apply:

e2eECC32_upper[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0]

e2eECC32_lower[7:0] = ecc_addr[7:0] ^ ecc_data_lower[7:0]

As a result, the PD's H-matrix is used for the BD RAM's e2eECC32 scheme with the appropriate 32 bit data word zeroed out:

e2eECC32_upper[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0]; data_lower[31:00] = 0

e2eECC32_lower[7:0] = ecc_addr[7:0] ^ ecc_data_lower[7:0]; data_upper[63:32] = 0

In any case, the resulting ECC32 is calculated based on the appropriate 32 data bits plus 29 address bits (the upper bits of the 32-bit address field).

The complete H-matrix for this (101, 93) code is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'd to form the final checkbit value on the left. For 32-bit data writes, the table sections corresponding to D[63:32] or D[31:0], together with A[31:3] are logically summed (output of each table section is XOR'd) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 2-1. e2eECC Basic H-Matrix Definition

Checkbits [7:0]	Data Bit[1]																															
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
7	*				*					*			*		*	*		*	*	*	*					*	*			*	*	

Table continues on the next page...

Table 2-1. e2eECC Basic H-Matrix Definition (continued)

6			*	*			*	*					*		*		*				*				*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*			
4	*		*			*		*					*		*		*			*			*	*			*				*	
3									*	*	*	*	*		*				*		*				*	*	*	*				
2						*		*					*		*			*	*				*	*							*	
1			*		*		*		*	*	*	*	*						*		*			*							*	
0			*			*	*	*				*			*	*	*			*				*				*			*	*	
			Byte 3							**Byte 2**								**Byte 1**								**Byte 0**						
Checkbits [7:0]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
7					*		*				*		*				*	*			*	*			*				*	*		*
6	*	*		*	*				*				*				*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*		*								
4	*		*	*				*		*	*				*	*	*			*			*	*				*		*		
3	*	*	*			*						*	*		*	*			*				*		*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
1		*						*		*			*	*	*			*				*				*			*	*	*	
0		*		*		*		*				*				*				*				*				*	*		*	*
Checkbits [7:0]	**Address Bit[1]**																															
	31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**			
7		*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*				
6		*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
5		*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
3	*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*				
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*				
1	*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*				
0	*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*				

1. Bit numbering is AHB convention, bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

Figure 2-1 through Figure 2-4 show an alternative representation of the ECC32 encode process, written as a C language function.

```
encodeEcc32_upper (addr, data_a2_is_zero)
    unsigned int        addr;                   /* 32-bit byte address */
    unsigned int        data_a2_is_zero;        /* 32-bit data upper, a[2]=0 */

{
    unsigned int        addr_ecc;               /* 8 bits of ecc for address */
    unsigned int        ecc;                    /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the bit = 1 */

    addr_ecc
        = (((addr           >> 31) & 1) ? 0x1f : 0x0)       /* addr[31] */
        ^ (((addr           >> 30) & 1) ? 0xf4 : 0x0)       /* addr[30] */
        ^ (((addr           >> 29) & 1) ? 0x3b : 0x0)       /* addr[29] */
        ^ (((addr           >> 28) & 1) ? 0xe3 : 0x0)       /* addr[28] */
        ^ (((addr           >> 27) & 1) ? 0x5d : 0x0)       /* addr[27] */
        ^ (((addr           >> 26) & 1) ? 0xda : 0x0)       /* addr[26] */
        ^ (((addr           >> 25) & 1) ? 0x6e : 0x0)       /* addr[25] */
        ^ (((addr           >> 24) & 1) ? 0xb5 : 0x0)       /* addr[24] */

        ^ (((addr           >> 23) & 1) ? 0x8f : 0x0)       /* addr[23] */
        ^ (((addr           >> 22) & 1) ? 0xd6 : 0x0)       /* addr[22] */
        ^ (((addr           >> 21) & 1) ? 0x79 : 0x0)       /* addr[21] */
        ^ (((addr           >> 20) & 1) ? 0xba : 0x0)       /* addr[20] */
        ^ (((addr           >> 19) & 1) ? 0x9b : 0x0)       /* addr[19] */
        ^ (((addr           >> 18) & 1) ? 0xe5 : 0x0)       /* addr[18] */
        ^ (((addr           >> 17) & 1) ? 0x57 : 0x0)       /* addr[17] */
        ^ (((addr           >> 16) & 1) ? 0xec : 0x0)       /* addr[16] */

        ^ (((addr           >> 15) & 1) ? 0xc7 : 0x0)       /* addr[15] */
        ^ (((addr           >> 14) & 1) ? 0xae : 0x0)       /* addr[14] */
        ^ (((addr           >> 13) & 1) ? 0x67 : 0x0)       /* addr[13] */
        ^ (((addr           >> 12) & 1) ? 0x9d : 0x0)       /* addr[12] */
        ^ (((addr           >> 11) & 1) ? 0x5b : 0x0)       /* addr[11] */
        ^ (((addr           >> 10) & 1) ? 0xe6 : 0x0)       /* addr[10] */
        ^ (((addr           >>  9) & 1) ? 0x3e : 0x0)       /* addr[ 9] */
        ^ (((addr           >>  8) & 1) ? 0xf1 : 0x0)       /* addr[ 8] */

        ^ (((addr           >>  7) & 1) ? 0xdc : 0x0)       /* addr[ 7] */
        ^ (((addr           >>  6) & 1) ? 0xe9 : 0x0)       /* addr[ 6] */
        ^ (((addr           >>  5) & 1) ? 0x3d : 0x0)       /* addr[ 5] */
        ^ (((addr           >>  4) & 1) ? 0xf2 : 0x0)       /* addr[ 4] */
        ^ (((addr           >>  3) & 1) ? 0x2f : 0x0);      /* addr[ 3] */
```

Figure 2-1. C language encodeECC32_upper function description (Part 1 of 4)

```
    ecc = (((data_a2_is_zero >> 31) & 1) ? 0xb0 : 0x0)      /* data[63] */
          ^ (((data_a2_is_zero >> 30) & 1) ? 0x23 : 0x0)      /* data[62] */
          ^ (((data_a2_is_zero >> 29) & 1) ? 0x70 : 0x0)      /* data[61] */
          ^ (((data_a2_is_zero >> 28) & 1) ? 0x62 : 0x0)      /* data[60] */
          ^ (((data_a2_is_zero >> 27) & 1) ? 0x85 : 0x0)      /* data[59] */
          ^ (((data_a2_is_zero >> 26) & 1) ? 0x13 : 0x0)      /* data[58] */
          ^ (((data_a2_is_zero >> 25) & 1) ? 0x45 : 0x0)      /* data[57] */
          ^ (((data_a2_is_zero >> 24) & 1) ? 0x52 : 0x0)      /* data[56] */

          ^ (((data_a2_is_zero >> 23) & 1) ? 0x2a : 0x0)      /* data[55] */
          ^ (((data_a2_is_zero >> 22) & 1) ? 0x8a : 0x0)      /* data[54] */
          ^ (((data_a2_is_zero >> 21) & 1) ? 0x0b : 0x0)      /* data[53] */
          ^ (((data_a2_is_zero >> 20) & 1) ? 0x0e : 0x0)      /* data[52] */
          ^ (((data_a2_is_zero >> 19) & 1) ? 0xf8 : 0x0)      /* data[51] */
          ^ (((data_a2_is_zero >> 18) & 1) ? 0x25 : 0x0)      /* data[50] */
          ^ (((data_a2_is_zero >> 17) & 1) ? 0xd9 : 0x0)      /* data[49] */
          ^ (((data_a2_is_zero >> 16) & 1) ? 0xa1 : 0x0)      /* data[48] */

          ^ (((data_a2_is_zero >> 15) & 1) ? 0x54 : 0x0)      /* data[47] */
          ^ (((data_a2_is_zero >> 14) & 1) ? 0xa7 : 0x0)      /* data[46] */
          ^ (((data_a2_is_zero >> 13) & 1) ? 0xa8 : 0x0)      /* data[45] */
          ^ (((data_a2_is_zero >> 12) & 1) ? 0x92 : 0x0)      /* data[44] */
          ^ (((data_a2_is_zero >> 11) & 1) ? 0xc8 : 0x0)      /* data[43] */
          ^ (((data_a2_is_zero >> 10) & 1) ? 0x07 : 0x0)      /* data[42] */
          ^ (((data_a2_is_zero >>  9) & 1) ? 0x34 : 0x0)      /* data[41] */
          ^ (((data_a2_is_zero >>  8) & 1) ? 0x32 : 0x0)      /* data[40] */

          ^ (((data_a2_is_zero >>  7) & 1) ? 0x68 : 0x0)      /* data[39] */
          ^ (((data_a2_is_zero >>  6) & 1) ? 0x89 : 0x0)      /* data[38] */
          ^ (((data_a2_is_zero >>  5) & 1) ? 0x98 : 0x0)      /* data[37] */
          ^ (((data_a2_is_zero >>  4) & 1) ? 0x49 : 0x0)      /* data[36] */
          ^ (((data_a2_is_zero >>  3) & 1) ? 0x61 : 0x0)      /* data[35] */
          ^ (((data_a2_is_zero >>  2) & 1) ? 0x86 : 0x0)      /* data[34] */
          ^ (((data_a2_is_zero >>  1) & 1) ? 0x91 : 0x0)      /* data[33] */
          ^  ((data_a2_is_zero        & 1) ? 0x46 : 0x0);     /* data[32] */

    return(ecc);
    }
```

Figure 2-2. C language encodeECC32_upper function description (Part 2 of 4)

```
C Language encodeECC32_lower Function Description
encodeEcc32_lower (addr, data_a2_is_one)
    unsigned int        addr;                   /* 32-bit byte address */
    unsigned int        data_a2_is_one;         /* 32-bit data lower, a[2]=1 */

{
    unsigned int        addr_ecc;               /* 8 bits of ecc for address */
    unsigned int        ecc;                    /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the bit = 1 */

    addr_ecc
        = (((addr           >> 31) & 1) ? 0x1f : 0x0)       /* addr[31] */
        ^ (((addr           >> 30) & 1) ? 0xf4 : 0x0)       /* addr[30] */
        ^ (((addr           >> 29) & 1) ? 0x3b : 0x0)       /* addr[29] */
        ^ (((addr           >> 28) & 1) ? 0xe3 : 0x0)       /* addr[28] */
        ^ (((addr           >> 27) & 1) ? 0x5d : 0x0)       /* addr[27] */
        ^ (((addr           >> 26) & 1) ? 0xda : 0x0)       /* addr[26] */
        ^ (((addr           >> 25) & 1) ? 0x6e : 0x0)       /* addr[25] */
        ^ (((addr           >> 24) & 1) ? 0xb5 : 0x0)       /* addr[24] */

        ^ (((addr           >> 23) & 1) ? 0x8f : 0x0)       /* addr[23] */
        ^ (((addr           >> 22) & 1) ? 0xd6 : 0x0)       /* addr[22] */
        ^ (((addr           >> 21) & 1) ? 0x79 : 0x0)       /* addr[21] */
        ^ (((addr           >> 20) & 1) ? 0xba : 0x0)       /* addr[20] */
        ^ (((addr           >> 19) & 1) ? 0x9b : 0x0)       /* addr[19] */
        ^ (((addr           >> 18) & 1) ? 0xe5 : 0x0)       /* addr[18] */
        ^ (((addr           >> 17) & 1) ? 0x57 : 0x0)       /* addr[17] */
        ^ (((addr           >> 16) & 1) ? 0xec : 0x0)       /* addr[16] */

        ^ (((addr           >> 15) & 1) ? 0xc7 : 0x0)       /* addr[15] */
        ^ (((addr           >> 14) & 1) ? 0xae : 0x0)       /* addr[14] */
        ^ (((addr           >> 13) & 1) ? 0x67 : 0x0)       /* addr[13] */
        ^ (((addr           >> 12) & 1) ? 0x9d : 0x0)       /* addr[12] */
        ^ (((addr           >> 11) & 1) ? 0x5b : 0x0)       /* addr[11] */
        ^ (((addr           >> 10) & 1) ? 0xe6 : 0x0)       /* addr[10] */
        ^ (((addr           >>  9) & 1) ? 0x3e : 0x0)       /* addr[ 9] */
        ^ (((addr           >>  8) & 1) ? 0xf1 : 0x0)       /* addr[ 8] */

        ^ (((addr           >>  7) & 1) ? 0xdc : 0x0)       /* addr[ 7] */
        ^ (((addr           >>  6) & 1) ? 0xe9 : 0x0)       /* addr[ 6] */
        ^ (((addr           >>  5) & 1) ? 0x3d : 0x0)       /* addr[ 5] */
        ^ (((addr           >>  4) & 1) ? 0xf2 : 0x0)       /* addr[ 4] */
        ^ (((addr           >>  3) & 1) ? 0x2f : 0x0);      /* addr[ 3] */
```

Figure 2-3. C language encodeECC32_upper function description (Part 3 of 4)

```
        ecc = (((data_a2_is_one  >> 31) & 1) ? 0x58 : 0x0)      /* data[31] */
            ^ (((data_a2_is_one  >> 30) & 1) ? 0x4f : 0x0)      /* data[30] */
            ^ (((data_a2_is_one  >> 29) & 1) ? 0x38 : 0x0)      /* data[29] */
            ^ (((data_a2_is_one  >> 28) & 1) ? 0x75 : 0x0)      /* data[28] */
            ^ (((data_a2_is_one  >> 27) & 1) ? 0xc4 : 0x0)      /* data[27] */
            ^ (((data_a2_is_one  >> 26) & 1) ? 0x0d : 0x0)      /* data[26] */
            ^ (((data_a2_is_one  >> 25) & 1) ? 0xa4 : 0x0)      /* data[25] */
            ^ (((data_a2_is_one  >> 24) & 1) ? 0x37 : 0x0)      /* data[24] */

            ^ (((data_a2_is_one  >> 23) & 1) ? 0x64 : 0x0)      /* data[23] */
            ^ (((data_a2_is_one  >> 22) & 1) ? 0x16 : 0x0)      /* data[22] */
            ^ (((data_a2_is_one  >> 21) & 1) ? 0x94 : 0x0)      /* data[21] */
            ^ (((data_a2_is_one  >> 20) & 1) ? 0x29 : 0x0)      /* data[20] */
            ^ (((data_a2_is_one  >> 19) & 1) ? 0xea : 0x0)      /* data[19] */
            ^ (((data_a2_is_one  >> 18) & 1) ? 0x26 : 0x0)      /* data[18] */
            ^ (((data_a2_is_one  >> 17) & 1) ? 0x1a : 0x0)      /* data[17] */
            ^ (((data_a2_is_one  >> 16) & 1) ? 0x19 : 0x0)      /* data[16] */

            ^ (((data_a2_is_one  >> 15) & 1) ? 0xd0 : 0x0)      /* data[15] */
            ^ (((data_a2_is_one  >> 14) & 1) ? 0xc2 : 0x0)      /* data[14] */
            ^ (((data_a2_is_one  >> 13) & 1) ? 0x2c : 0x0)      /* data[13] */
            ^ (((data_a2_is_one  >> 12) & 1) ? 0x51 : 0x0)      /* data[12] */
            ^ (((data_a2_is_one  >> 11) & 1) ? 0xe0 : 0x0)      /* data[11] */
            ^ (((data_a2_is_one  >> 10) & 1) ? 0xa2 : 0x0)      /* data[10] */
            ^ (((data_a2_is_one  >>  9) & 1) ? 0x1c : 0x0)      /* data[ 9] */
            ^ (((data_a2_is_one  >>  8) & 1) ? 0x31 : 0x0)      /* data[ 8] */

            ^ (((data_a2_is_one  >>  7) & 1) ? 0x8c : 0x0)      /* data[ 7] */
            ^ (((data_a2_is_one  >>  6) & 1) ? 0x4a : 0x0)      /* data[ 6] */
            ^ (((data_a2_is_one  >>  5) & 1) ? 0x4c : 0x0)      /* data[ 5] */
            ^ (((data_a2_is_one  >>  4) & 1) ? 0x15 : 0x0)      /* data[ 4] */
            ^ (((data_a2_is_one  >>  3) & 1) ? 0x83 : 0x0)      /* data[ 3] */
            ^ (((data_a2_is_one  >>  2) & 1) ? 0x9e : 0x0)      /* data[ 2] */
            ^ (((data_a2_is_one  >>  1) & 1) ? 0x43 : 0x0)      /* data[ 1] */
            ^  ((data_a2_is_one         & 1) ? 0xc1 : 0x0);     /* data[ 0] */

    return(ecc);
    }
```

Figure 2-4. C language encodeECC32_upper function description (Part 4 of 4)

As the ECC syndrome is calculated on a read operation by applying the H-matrix to the data plus the checkbits, an all zero syndrome indicates an error free operation. If the generated syndrome value is non-zero and matches one of the H-matrix values associated with the data or checkbits, it represents a single-bit error correction case and the specific bit is complemented to produce the correct data value. If the syndrome value matches one of the H-matrix values associated with the address bits, the opposite data word, or is an even weight value, or represents an unused odd weight value, a non-correctable ECC event has been detected and the appropriate error termination response is initiated.

Chapter 3
Buddy Device—System Integration Unit Lite (BD_SIUL2)

3.1 Introduction

The Buddy Device—System Integration Unit Lite2 (BD_SIUL2) provides control over the electrical pads of the buddy device (BD), as well as implementing various SoC functions that are not included in other modules.

The main functions supported by BD_SIUL2 are:

- Device identification
- Device status reporting
- Pad control and signal muxing

It should be noted that some other system level features are located in the BD DCI block including:

- Device identification available via the JTAG accessible Nexus Device ID (DID) register
- Reporting of reset status via the JTAG accessible reset status register
- Control of production device (PD) and BD reset via JTAG accessible reset control register

These registers are also indirectly accessible by software via BD_JTAGM in software debug mode.

3.2 Device identification

The BD_SIUL2 includes a device ID register (DIDR) that identifies the BD type and version, allowing a connected tool or development/calibration software running on a connected PD to determine what debug or calibration resources are available. The DIDR uses similar formatting to the MCU ID registers (MIDR1, MIDR2) included in the SIUL2 module.

This information is made available via the BD SIU rather than via the PD because it is either not available via the PD (for example, information on BD-specific resources, such as extended overlay RAM size) or because it is useful for the tool to be able to access the information before the PD is powered up.

Information available includes:

- An ID code that can be used to uniquely identify the BD type
- A version field that is changed for every major mask revision of the BD
- A parameterized field that indicates the amount of extended overlay RAM included
    - The field coding allows indication of up to 4 MB of RAM, with resolution of 256 KB steps

It should be noted that in addition to information available in the DIDR, a connected tool can also retrieve identification information from other sources including:

- The JTAG ID returned by the buddy DCI/JTAGC module, which is different than the JTAG ID returned by a standard PD DCI/JTAGC
- The MIDR register included in the PD SIUL2 module, which includes information on the PD device type, version and package type (including identification of emulation and debug device package variants)

3.3 Device status information

This section describes the devices status information.

3.3.1 Initialization status

The BD SIU includes a flag (BD_INIT) that can be used by the connected PD or any connected debug or calibration tools to track the status of the BD.

The main expected usage mode for this bit is to allow the Boot Assist Flash (BAF) or user calibration software on the PD to determine whether the BD has lost power since it was last initialized. A typical usage sequence for BD_INIT is shown below.

- At BD power on reset (POR), BD_INIT is cleared automatically
  - BD_INIT is not cleared by external BD reset input
- A calibration tool or calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration)
- The calibration tool or calibration software sets BD_INIT
- After PD POR, the BAF executes and reads BD_INIT
  - If BD_INIT is clear, the BD overlay is assumed to be uninitialized and is not used for calibration remap
  - If BD_INIT is set, the BD overlay is assumed to be initialized and can be used for calibration remap. (Other tests to verify data integrity may be used.)

BD_INIT is cleared by BD POR, and can be set or cleared via the debug interface or writes from the PD. BD_INIT is only used to detect full power down events resulting in a subsequent POR. It does not detect brown out drops in the supply voltage.

3.3.2 PD reset status

The IPS accessible BD_SIU_RSR register shows what stage of reset processing the connected PD is in, and whether the PD is powered down or not. This is functionally different from the software accessible registers in the PD reset generation module that indicate which reset source has caused the last PD reset.

3.4 Memory mapped registers

This table gives an overview of the BD_SIUL2 registers.

BD_SIUL2 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C80_0000	Device Identification Register 1 (BD_SIUL2_DIDR1)	32	R	57BD_0000h	3.4.1/32
C80_0004	Device Identification Register 2 (BD_SIUL2_DIDR2)	32	R	0003_0100h	3.4.2/33
C80_0100	Initialization Status Register (BD_SIUL2_ISR)	32	R/W	0000_0000h	3.4.3/34
C80_0104	Reset Status Register (BD_SIUL2_RSR)	32	R	0000_0000h	3.4.4/35

3.4.1 Device Identification Register 1 (BD_SIUL2_DIDR1)

The BD_SIUL2_DIDR1 contains the part number and mask number.

Address: C80_0000h base + 0h offset = C80_0000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM																Reserved								MASKNUM_ MAJOR				MASKNUM_ MINOR			
W																																
Reset	0	1	0	1	0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BD_SIUL2_DIDR1 field descriptions

Field	Description
0–15 PARTNUM	Device Part Number. Read-only, mask programmed part number of the device. Reads 0x57BD.
16–23 Reserved	This field is reserved.
24–27 MASKNUM_ MAJOR	Major Mask Revision Number. Read-only, mask programmed mask number of the device. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.
28–31 MASKNUM_ MINOR	Minor Mask Revision Number. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.

3.4.2 Device Identification Register 2 (BD_SIUL2_DIDR2)

The BD_SIUL2_DIDR2 contains a complement to the part number and amount of RAM available.

Address: C80_0000h base + 4h offset = C80_0004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												RAM_SIZE				PARTNUM								Reserved							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

BD_SIUL2_DIDR2 field descriptions

Field	Description
0–11 Reserved	This field is reserved.
12–15 RAM_SIZE	RAM Size. Read-only, indicates the amount of overlay RAM available in the device, with resolution of 256K byte steps 0000 0.25M byte 0001 0.5M byte 0010 0.75M byte 0011 1M byte 0100 1.25M byte 0101 1.5M byte 0110 1.75M byte 0111 2M byte 1000 2.25M byte 1001 2.5M byte 1010 2.75M byte 1011 3M byte 1100 3.25M byte 1101 3.5M byte 1110 3.75M byte 1111 4M byte
16–23 PARTNUM	Device Part Number. Read-only, mask programmed complement to the part number of the device. For the BD 1M byte part with part number 57BD1, BD_SIUL2_DIDR1.PARTNUM=57BD, and BD_SIUL2_DIDR2.PARTNUM=1. 01 1 (BD1M) 02 2 (BD2M)
24–31 Reserved	This field is reserved.

3.4.3 Initialization Status Register (BD_SIUL2_ISR)

The BD_SIUL2_ISR contains a flag that can be used by the PD or any connected debug or calibration tool to track the status of the BD. This register is reset by POR.

The main expected use case for this flag is to allow software on the PD to determine whether the BD has lost power since it was last initialized. A typical sequence is:

- At power on reset (POR), the flag is cleared automatically (not affected by software reset)
- A calibration tool or PD calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration)
- The calibration tool or PD calibration software sets the BD_INIT flag
- After PD POR, the PD boot software executes and reads BD_INIT
- If BD_INIT is clear, BD overlay is assumed to be uninitialized and is not used for calibration remap
- If BD_INIT is set, BD overlay is assumed to be initialized and can be used for calibration remap (other tests to verify data integrity may be used)

BD_INIT is cleared by BD POR, and can be set or cleared via writes from the PD or from the Nexus RWA.

Address: C80_0000h base + 100h offset = C80_0100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															BD_ INIT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

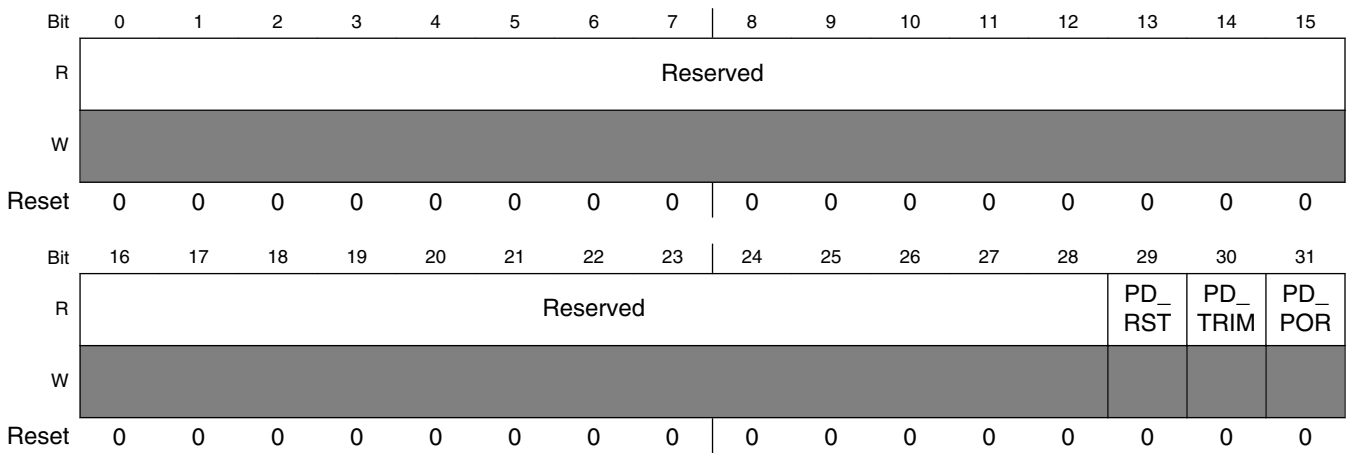
BD_SIUL2_ISR field descriptions

Field	Description
0–30 Reserved	This field is reserved.
31 BD_INIT	Initialization Flag. This is a read/write bit that can be used to indicate that the BD was initialized. Software reset does not affect the flag. Power-on reset clear the flag. 0 BD not initialized 1 BD initialized

3.4.4 Reset Status Register (BD_SIUL2_RSR)

The BD_SIUL2_RSR contains reset status information about the PD.

Address: C80_0000h base + 104h offset = C80_0104h



BD_SIUL2_RSR field descriptions

Field	Description
0–28 Reserved	This field is reserved.
29 PD_RST	PD Reset Flag. Read-only, indicates if PD has finished the reset sequence. 0 PD reset sequence not completed 1 PD reset sequence completed
30 PD_TRIM	PD Trim Flag. Read-only, indicates if PD has finished trimming phase during the reset sequence. 0 PD trimming phase not completed 1 PD trimming phase completed
31 PD_POR	PD POR Flag. Read-only, indicates whether PD is powered down or not. 0 PD is powered down or power on reset has not completed 1 PD is powered and power on reset has completed

Chapter 4
Buddy Device—Debug and Calibration Interface (DCI)

4.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard IEEE 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features in the Emulation and Debug Device (ED) that includes features in the Production device (PD) as well as features in the Buddy device (BD). There are two DCIs on the ED which are designated PD DCI and BD DCI.

4.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- Dual DCI system for multi-device clients
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- IEEE 1149.1 and IEEE 1149.7 controllers
- Debug break and cross-triggering control

- Synchronous restart control for CPU(s) when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

4.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Buddy device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.
    - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST's 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
    - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.
- I.JTAG— JTAG signals interconnecting the two DCI implementations in the ED coming from the intra-die pads.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as timers/ SPU, etc. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

This figure shows the DCI block diagram.

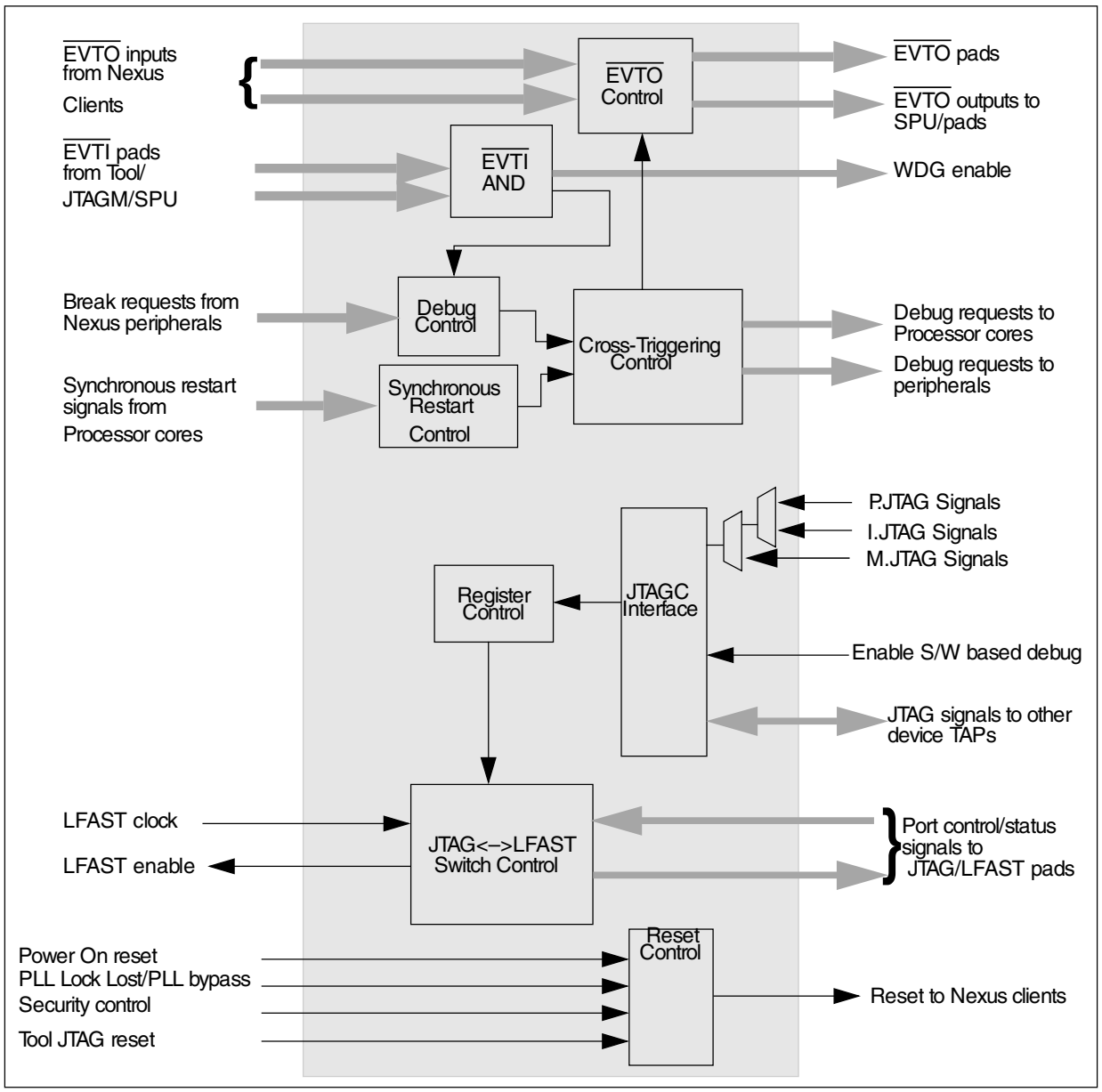


Figure 4-1. DCI block diagram

When the DCI is used in an ED package (PD and the BD together) its configuration is shown as in the following figure. In the ED package, a debug tool is connected to the BD pins and connects to the PD with the I.JTAG interface which are routed using inter-die pads.

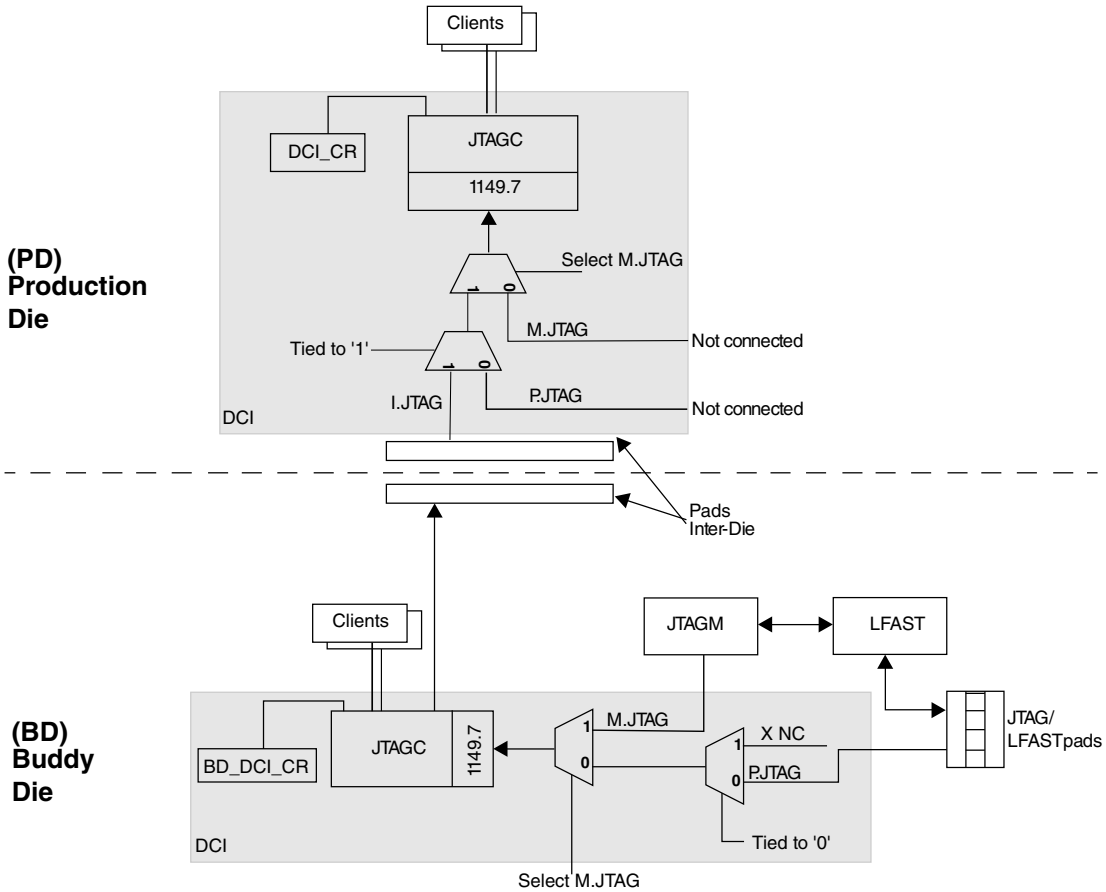


Figure 4-2. DCI in an emulation configuration

4.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

This table shows a summary of the configuration options.

Table 4-1. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI Source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated. In the ED, the BD DCI is reset with BD POR/Low voltage detection on BD_VDD.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the BD LFAST and BD JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed. Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in Port sharing between JTAG and LFAST modes. During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. Table 4-2 shows the port mapping.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 4-2. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

This figure shows the DCI mode switching operation.

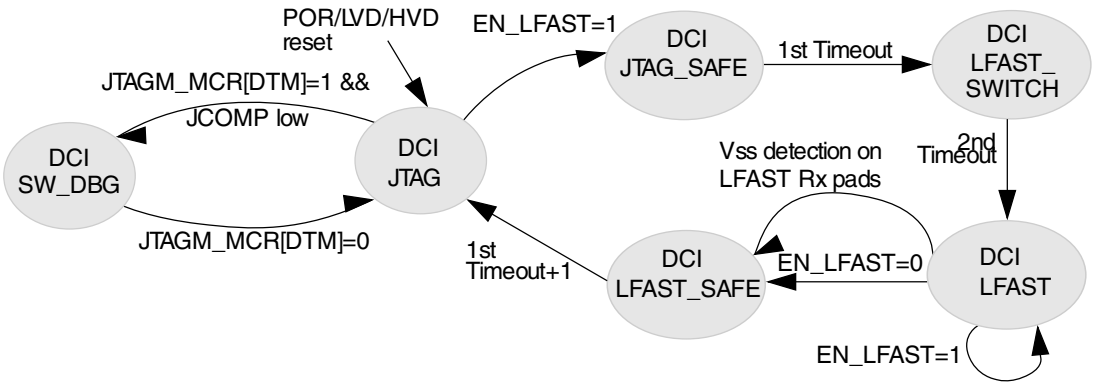


Figure 4-3. DCI operating modes state diagram

Note

JTAGM_MCR[jtagm_JCOMP] should not be asserted in LFAST mode to avoid any undesired behavior.

This table contains the operating mode descriptions.

Table 4-3. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG or I.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: • JCOMP is latched inside to control until switching gets completed • The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state • An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

This figure shows a block diagram of the JTAG/LFAST connections.

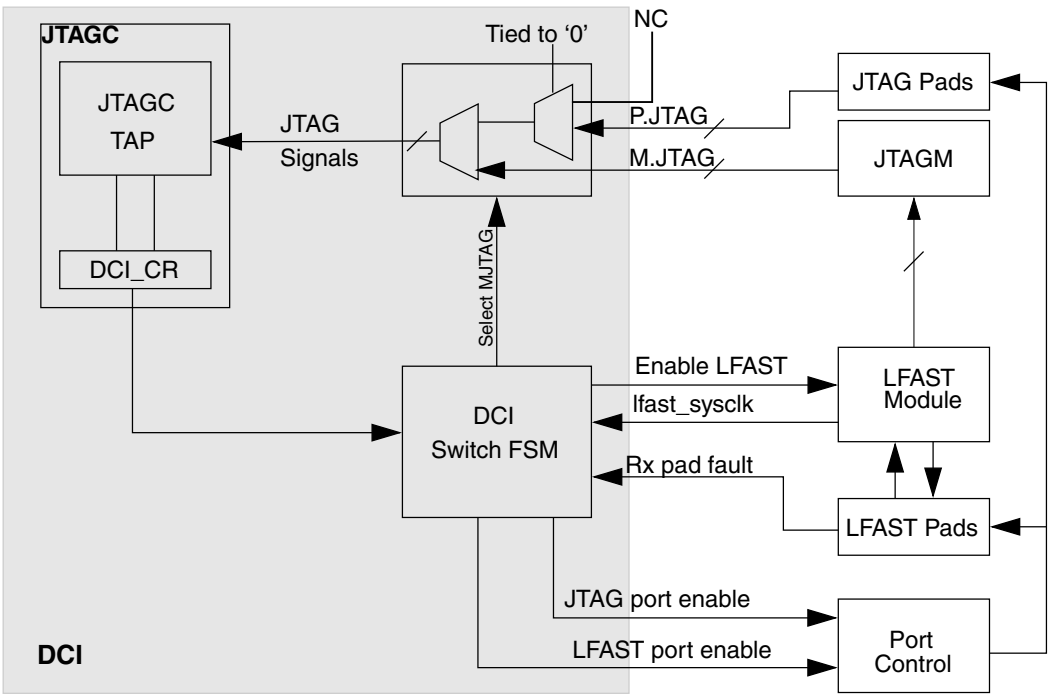


Figure 4-4. DCI JTAG/LFAST switch integration

4.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

- EN_LFAST—Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2—Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure.The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST

mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in Figure 4-5.

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

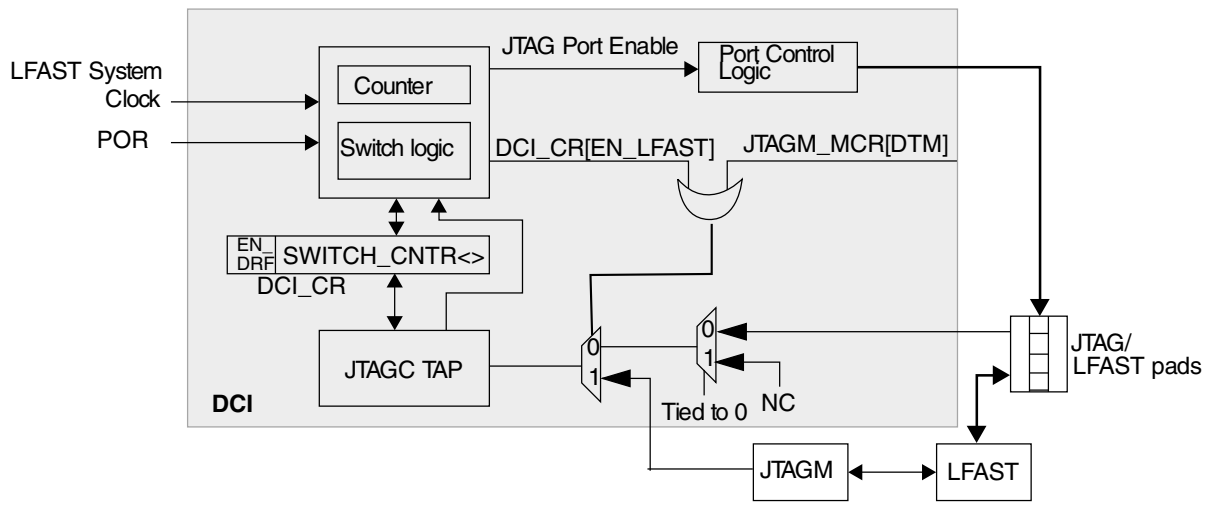


Figure 4-5. DCI switching between JTAG and LFAST modes

4.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
   - EN_LFAST
   - SWITCH_CNTR1
   - SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The

DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.

5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

This figure shows the timing sequence.

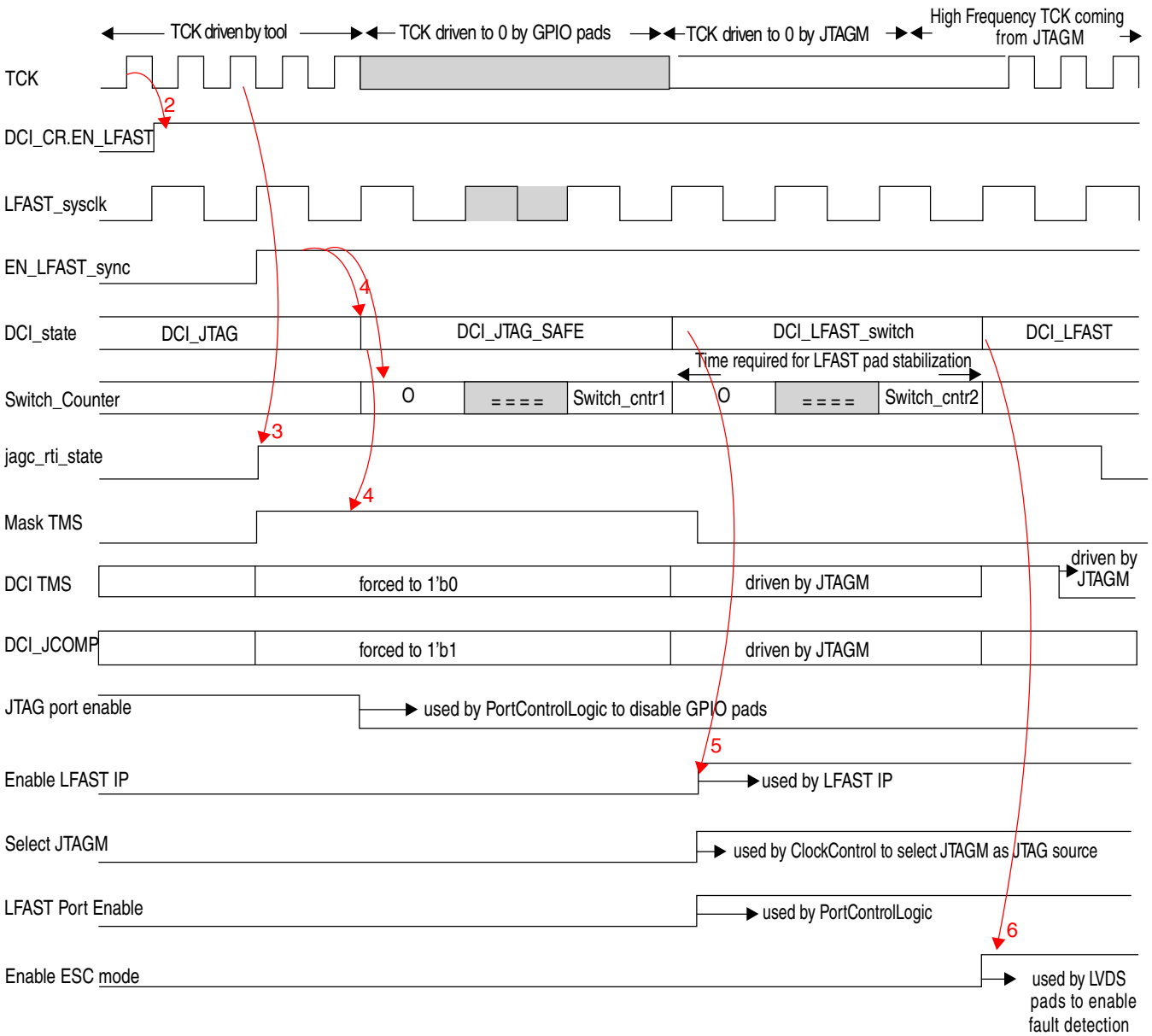


Figure 4-6. DCI JTAG to LFAST switching diagram

4.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.

3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

Table 4-7 and Table 4-9 summarize the MCU and tool behaviors. The following figure shows the timing sequence.

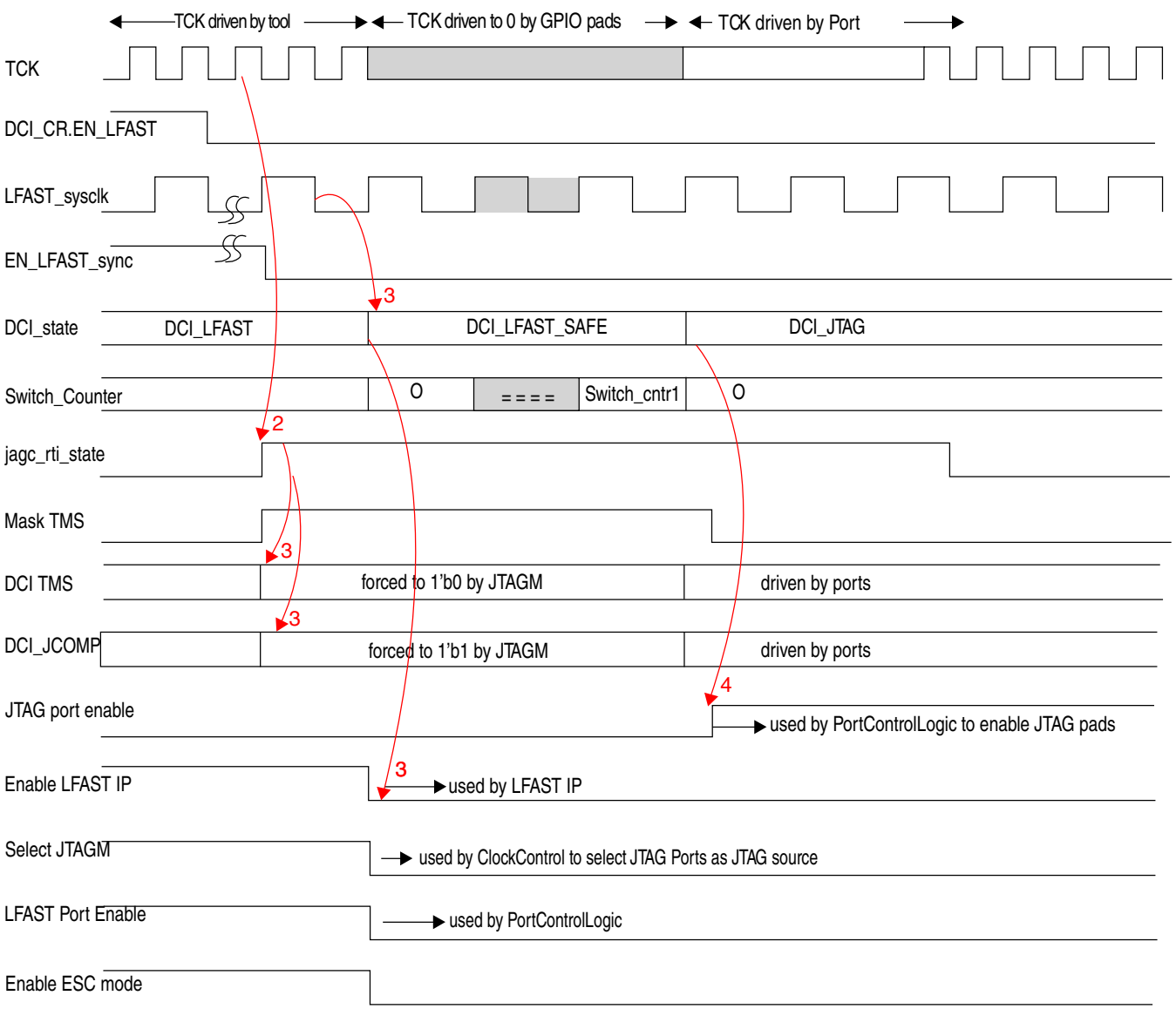


Figure 4-7. DCI LFAST to JTAG switching diagram

4.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

4.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

NOTE

It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

4.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks, with the PD DCI providing the reset signal to all PD debug blocks, and the BD DCI providing the reset signal for all BD debug blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect (applicable to both PD and BD)
- JCOMP (applicable to both PD and BD)
- SoC Debug_Enable signal (applicable to PD only)
- System PLL is out of lock state and not bypassed (applicable to PD only)

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the BD Nexus reset signal remains asserted until the JCOMP signal is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.

If Debug_Enable is de-asserted by the SoC security modules (HSM or PASS), then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 4-4. Nexus Reset Truth Table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert

Table continues on the next page...

Table 4-4. Nexus Reset Truth Table (continued)

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

4.1.3.5 System watchdog control

The BD DCI does not include the Software Watchdog Timer (SWT) disable feature.

4.1.3.6 Reset control

The BD DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force BD Reset—This reset is generated by programming DCI_CR[FBD_RESET] bit. When set it forces reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

4.1.3.7 Security

The SoC security logic provides a Debug_Enable signal to the PD DCI. The PD DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the PD DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

4.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

4.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. The BD DCI only contains the DCI_CR and does not contain the DCI_PINCR. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

4.3.1 DCI control register (BD_DCI_CR)

The BD_DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. The following table shows the format of the BD_DCI_CR.

NOTE

The BD version of this register is refered to as DCI_CR in the other sections of this chapter.

In an ED which has both PD and BD instances of the DCI, all bits controlling the switch between LFAST and JTAGM must be programmed on the BD DCI instance. The only bits of this register that do not follow this rule are EVTI1_PAD_EN and EVTI0_PAD_EN, which must be programmed in both DCI instances.

	31	30	29	28	27	26	25	24
R W	SWITCH_CNTR2							
Reset	1	1	1	1	1	1	1	1

	23	22	21	20	19	18	17	16
R W	SWITCH_CNTR2							
Reset	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
R W	SWITCH_CNTR1					FLVDS_PAD_FLT	Reserved	
Reset	1	1	1	1	1	0	0	0

	7	6	5	4	3	2	1	0
R W	FBD_RESET	FORCE_LFAST	EVTI_1_PAD_EN	EVTI_0_PAD_EN	Reserved		EN_ESC_MODE	EN_LFAST
Reset	0	0	0	0	0	0	0	0

The BD_DCI_CR is described in this table.

Table 4-5. BD_DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2. Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = 32,768×T_{lfast_sysclk}
15:11 SWITCH_CNTR1	Width of Switch_CNTR1. Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = 32×T_{lfast_sysclk}
10 FLVDS_PAD_FLT	Force LVDS Pad Fault. Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
7 FBD_RESET	Force Reset 0 Do not force Reset 1 Force Reset
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
5	$\overline{\text{EVTI}}$[1] feature enable

Table continues on the next page...

Table 4-5. BD_DCI_CR field descriptions (continued)

Field	Description	
EVTI1_PAD_EN[1]	0	$\overline{\text{EVTI}}$[1] disabled (default)
	1	$\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN[1]	$\overline{\text{EVTI}}$[0] feature enable	
	0	$\overline{\text{EVTI}}$[0] disabled (default)
	1	$\overline{\text{EVTI}}$[0] enabled
1 EN_ESC_MODE	Enable Escape Mode. Fault detect feature enable for LFAST Rx pads	
	0	Fault detect feature not present for LFAST Rx pads (default)
	1	Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST[2]	Enable for LFAST-based tool selection. This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset.	
	0	LFAST tool not enabled. DCI operates in JTAG mode (default)
	1	LFAST tool enabled

1. In an ED that has both PD and BD, EVTI1_PAD_EN and EVTI0_PAD_EN must be programmed in both BD and PD instances of the DCI in order to allow events generated on the BD to pass through the BD DCI and reach the PD DCI.
2. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

4.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

4.4 Functional description

This section describes the functional description of the DCI.

4.4.1 DCI mode transition

After power-on reset, the BD DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. Table 4-6 and Table 4-7 show the steps and expected

states for the MCU and tool for JTAG to LFAST switching. Table 4-8 and Table 4-9 contains the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 4-6. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_ SYSCLK _EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 4-7. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode)[1]	DCI_CR[EN_ LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 4-8. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 4-9. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

4.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the BD LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the BD DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode.Table 4-10 and Table 4-11 contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 4-10. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input
Mode Escape	Enable	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 4-11. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don't Care
TMS	LVDS TxP	Don't Care
TDO	LVDS RxP	Hi-Z

Table continues on the next page...

Table 4-11. Tool operation (LFAST to JTAGRST switching) (continued)

Tool	LFAST mode	Tool forces JCOMP low
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don't Care

Chapter 5
Buddy Device—JTAG Controller (JTAGC)

5.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

5.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

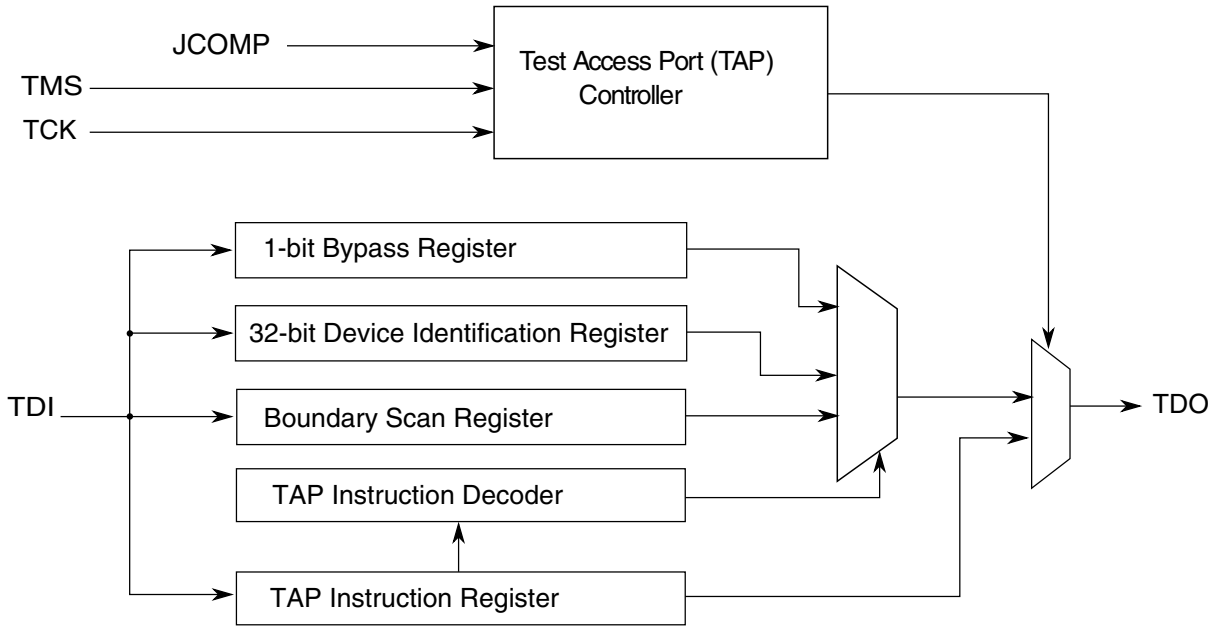


Figure 5-1. JTAG (IEEE 1149.1) block diagram

5.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 5-4 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Bypass register, boundary scan register, data registers, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

5.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

5.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

5.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

5.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

5.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 5-1. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z[1]	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

5.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

5.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

5.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

5.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

5.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

5.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

5.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

Figure 5-2. Instruction register

5.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

5.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
Reset	PRN				DC						PIN					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code											1
W																
Reset	PIN (contd.)				MIC											1

The following table describes the device identification register functions.

Table 5-2. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x2B.
PIN	Part Identification Number. Contains the part number of the device. 0x381.

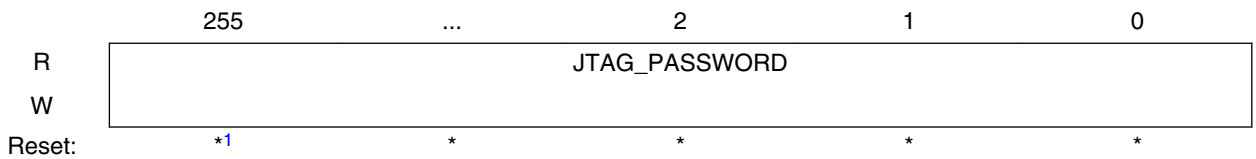
Table continues on the next page...

Table 5-2. Device identification register field descriptions (continued)

Field	Description
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E0x020.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

5.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256 'b0.

1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 5-3. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

5.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary

scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

The BD boundary scan chain includes registers connected to all PD/BD interface signals at the BD side. These registers are controlled by the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions applied to the BD JTAG controller. Similarly, the PD boundary scan chain also contains registers connected to the interface signals at the PD side, which are controlled by instructions applied to the PD JTAG controller.

5.4 Functional description

This section explains the JTAGC functional description.

5.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

5.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

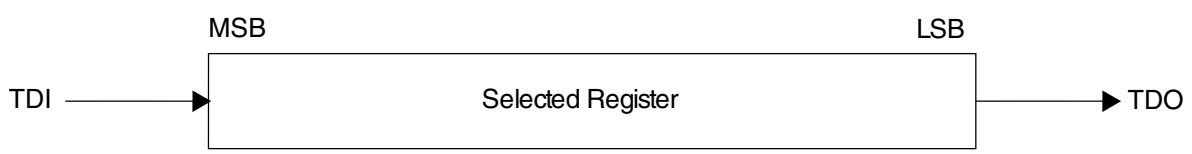
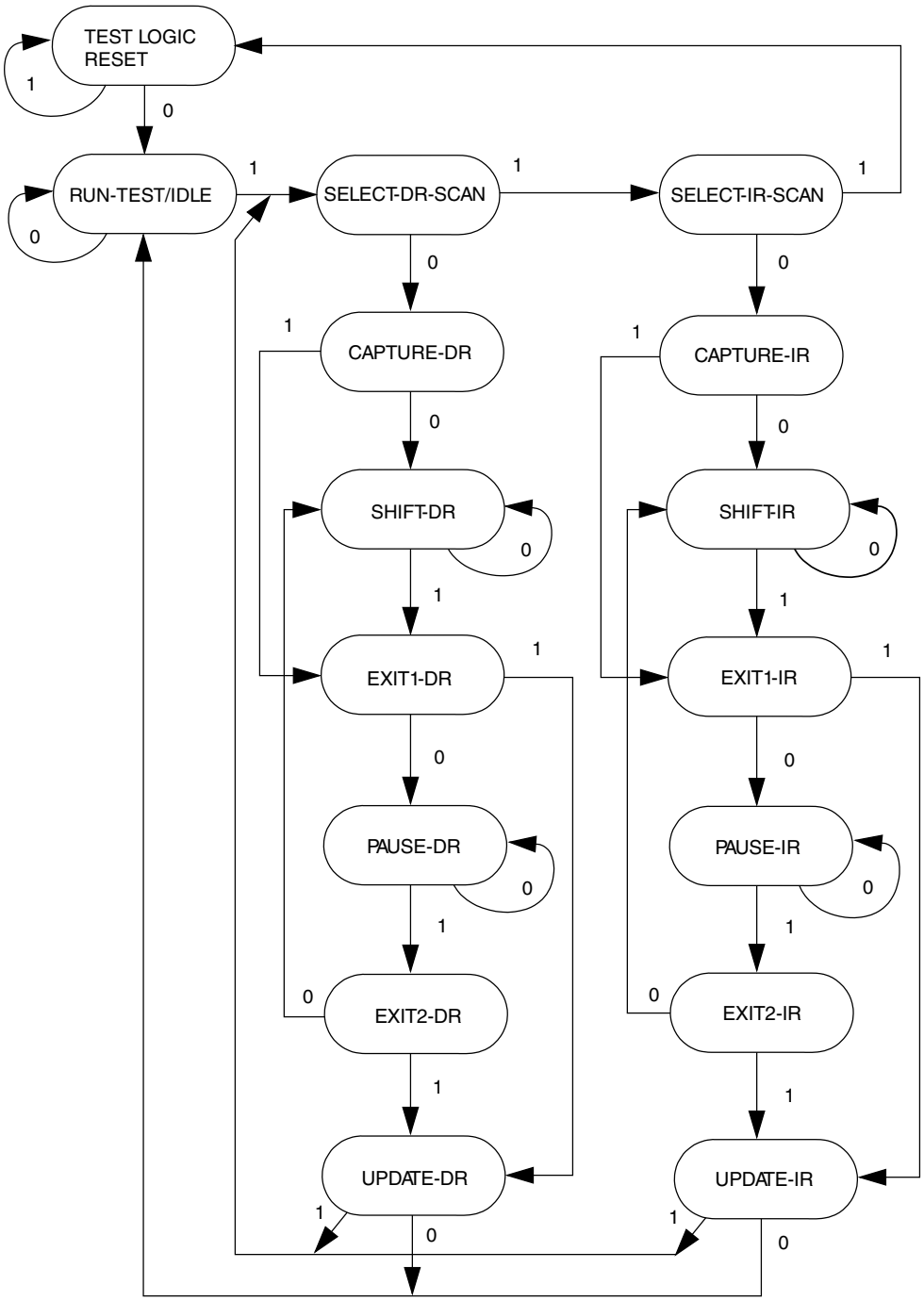


Figure 5-3. Shifting data through a register

5.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 5-4. IEEE 1149.1-2001 TAP controller finite state machine

5.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

5.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

5.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 5-4. General 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
CLAMP	001100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
ENABLE_DCI_CR	001110	Enables access to DCI CR register

Table continues on the next page...

Table 5-4. General 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	001111	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

5.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

5.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

5.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

5.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

5.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

5.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

5.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a

single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

5.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

5.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

5.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

PD and the BD have their own separate boundary scan chains, each one controlled by the respective JTAG controller. Besides the package I/O pins, these chains also encompasses the interface signals between BD and PD. Testing of the interface can be done by applying EXTEST, SAMPLE or SAMPLE/PRELOAD instructions to both PD and BD JTAG.

5.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

Chapter 6
Buddy Device—JTAG Master (JTAGM)

6.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

6.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

- Configuration register
- Status register
- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

This figure shows the block diagram of the JTAGM.

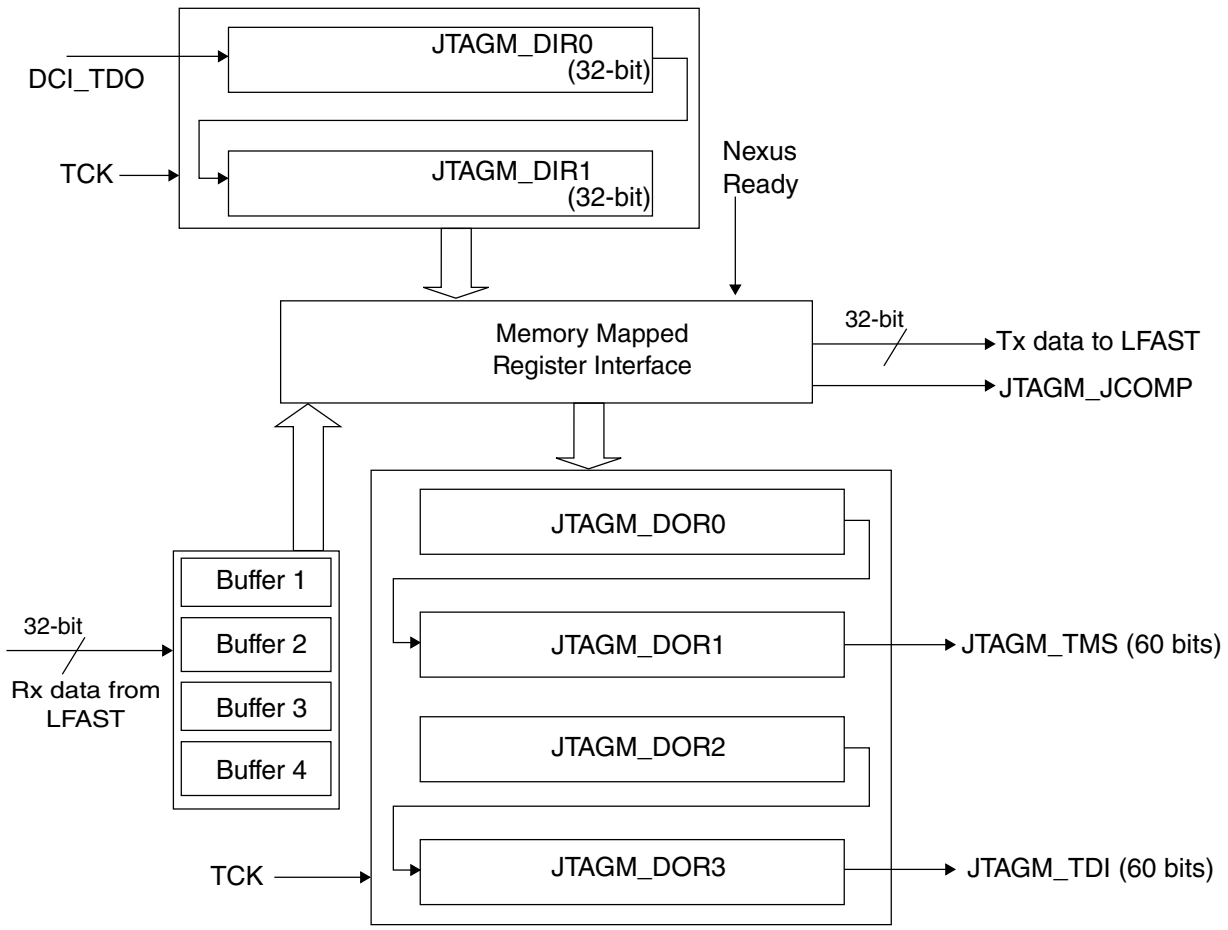


Figure 6-1. JTAGM block diagram

6.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

6.2 Functional description

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

This figure shows the flow diagram of the LFAST to DCI connection.

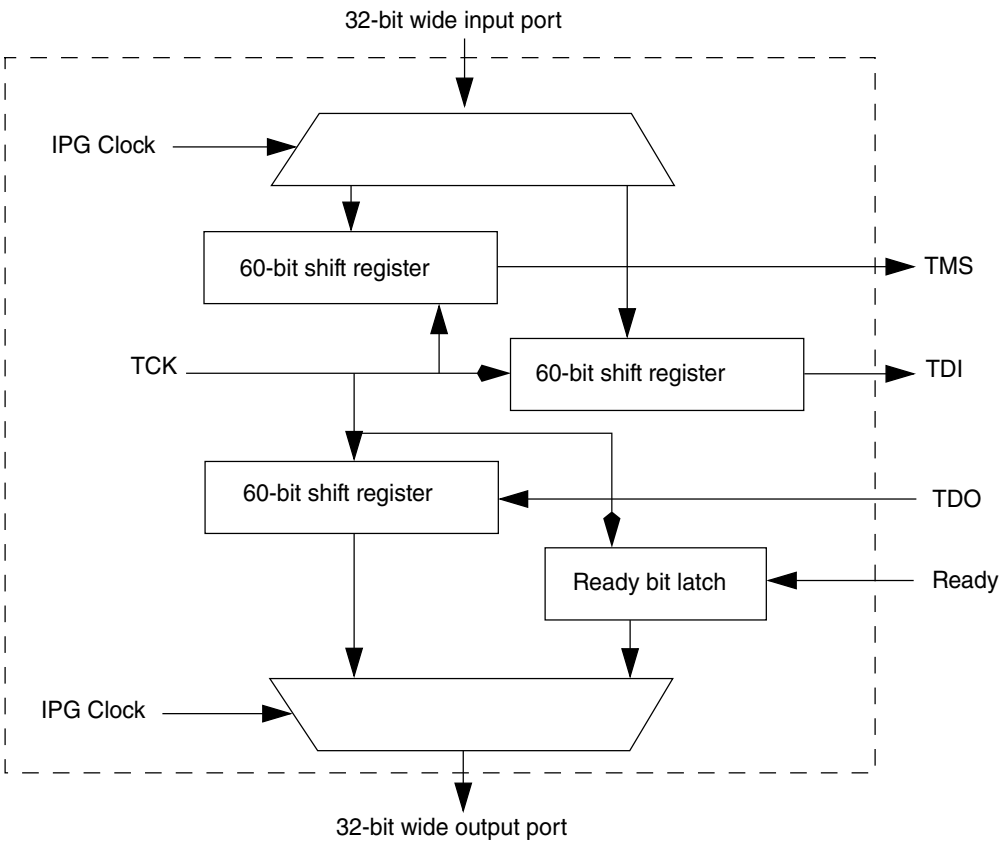


Figure 6-2. LFAST-to-DCI flow diagram

6.2.1 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAGM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
  - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
  - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

NOTE

The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

6.2.2 JTAGM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAGM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAGM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAGM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAGM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAGM status register (JTAGM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAGM to the LFAST also includes an 8-bit CRC. The data sent from the JTAGM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the

first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

6.2.3 JTAGM message tagging

The JTAGM, in conjunction with the LFAST, is pipelined to allow several JTAGM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAGM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAGM returns the response on channel E; if the tool sends a data command on channel F, the JTAGM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAGM are sent on channel A and response data is also sent back on channel A.

6.2.4 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

6.2.4.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

6.2.4.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

6.2.5 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus $\overline{\text{EVTI}}$ (Event In) and $\overline{\text{EVTO}}$ (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ signals to be embedded into the serial protocol. The $\overline{\text{EVTI}}$0 and $\overline{\text{EVTI}}$1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either $\overline{\text{EVTO}}$0 or $\overline{\text{EVTO}}$1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI.

The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

6.2.6 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in Status register ready bit). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written +
- 32-bit value to be written to the configuration register +
- 24-bit mask of which bits within the status register should be written +
- 8 bits of zero-padding +
- 24-bits of data to be written to the status register +
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data).

The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module.

The JTAGM requests data from the LFAST only when at least one internal buffer is free.

6.2.7 JTAGM to DCI serial interface

This figure shows the data transfer timings between JTAGM and DCI.

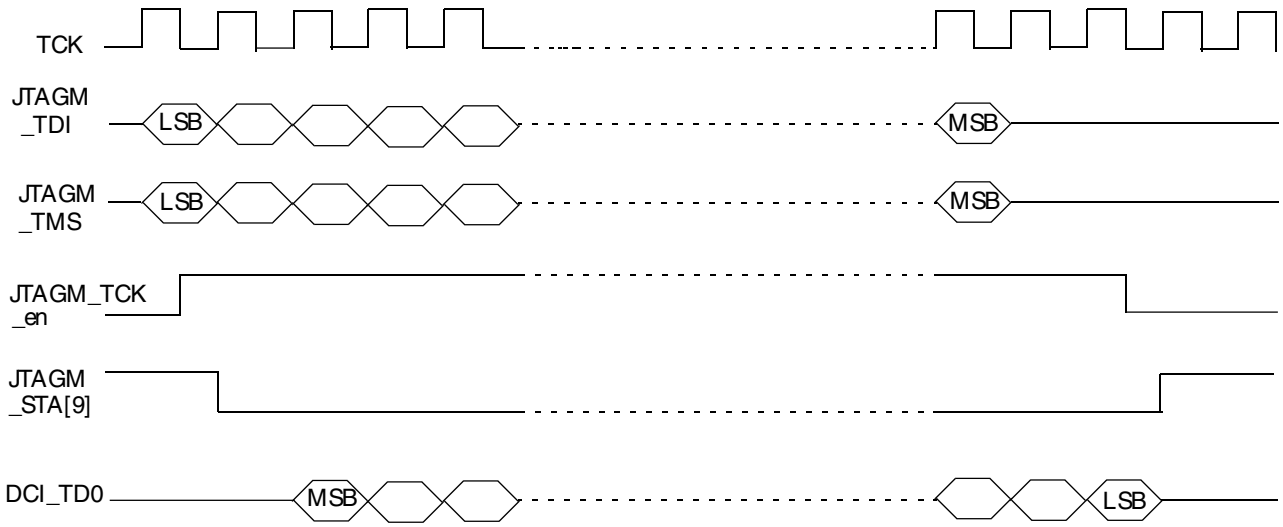


Figure 6-3. JTAGM to DCI serial interface diagram

6.2.8 JTAGM data handling and CRC calculation in LFAST mode

This figure shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

bit 127 to bit 68		bit 67 to bit 8		7	6	5	4	3	2	1	0
TMS (60 bits)		TDI (60 bits)		CRC (8 bits)							
bit 0 to bit 59		bit 0 to bit 59									
Last TMS bit transmitted by JTAGM	First TMS bit transmitted by JTAGM	Last TDI bit transmitted by JTAGM	First TDI bit transmitted by JTAGM								

Figure 6-4. Bit ordering for LFAST to JTAGM transfer

This figure shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

bit 95 to bit 36		bit 35 to bit 32	bit 31 to bit 8	7	6	5	4	3	2	1	0
TDO (60 bits)		0 pad (4 bits)	JTAGM_SR (24 bits)	CRC (8 bits)							
bit 0 to bit 59		bit 0 to bit 59	bit 31 to bit 8								
First TDO bit received by JTAGM	Last TDO bit received by JTAGM										

Figure 6-5. Bit ordering for JTAGM to LFAST transfer

The CRC calculation is described in following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in Figure 6-5).

6.2.9 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

6.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

6.4 Memory mapped registers

JTAGM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JTAGM_MCR)	32	R/W	0300_0002h	6.4.1/83
4	Status Register (JTAGM_SR)	32	R/W	0020_0200h	6.4.2/85
8	Data Out Register 0 (JTAGM_DOR0)	32	R/W	0000_0000h	6.4.3/88
C	Data Out Register 1 (JTAGM_DOR1)	32	R/W	0000_0000h	6.4.4/88
10	Data Out Register 2 (JTAGM_DOR2)	32	R/W	0000_0000h	6.4.5/88
14	Data Out Register 3 (JTAGM_DOR3)	32	R/W	0000_0000h	6.4.6/89
18	Receive CRC Register (JTAGM_RxCRC)	32	R	0000_0000h	6.4.7/89
1C	Data Input Register 0 (JTAGM_DIR0)	32	R	0000_0000h	6.4.8/90
20	Data Input Register 1 (JTAGM_DIR1)	32	R	0000_0000h	6.4.9/90

6.4.1 Module Configuration Register (JTAGM_MCR)

This register contains the status bits for the current transfer. In the Emulation and Debug Device (ED), all bits of this register need to be programmed in the Buddy Device instance of the JTAGM. (The only exception is the SIE bit, which is only seen in the PD instance of the JTAGM and is therfore not seen in this chapter. The SEI bit always needs to be programmed in the PD instance of the JTAGM, regardless of being ED or not.)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWRESET								Reserved							
W		evto0_sense		evto1_sense		evto_IE	evti0_assert	evti1_assert								
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		inter_jtag_frame_timer						Reserved		IIE	TCKSEL			jtagm_JCOMP	DTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

JTAGM_MCR field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit; after being written with a 1, this bit is auto-cleared within 3 TCK cycles. Writing a 0 has no effect.
1–2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO0}}$ 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$ 0 edges (default setting)

Table continues on the next page...

JTAGM_MCR field descriptions (continued)

Field	Description
	01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$0 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$0 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$0
3–4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$ 1 line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$ 1 edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$ 1 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$ 1 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$ 1
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt. 0 No JTAGM interrupt is generated In response to any $\overline{\text{EVTO}}$ 0 or $\overline{\text{EVTO}}$ 1 activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	$\overline{\text{EVTI}}$0 Assert. 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	$\overline{\text{EVTI}}$1 Assert. In software mode (DTM = 1), the internal $\overline{\text{EVTI}}$ signal remains de-asserted, regardless of the state of this bit and writing to this bit has no effect. 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted
8–17 Reserved	This field is reserved.
18–23 inter_jtag_frame_timer	TCK delay. 000000 0 TCK delay 000001 1 TCK delay ... ... 111111 63 TCK delay
24–25 Reserved	This field is reserved.
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27–29 TCKSEL	TCK clock frequency selection. When the JTAGM is enabled, this bit should not be programmed with 000. 000 Reserved 001 TCK is system clock ÷ 2 010 TCK is system clock ÷ 3 011 TCK is system clock ÷ 4 100 TCK is system clock ÷ 5 101 TCK is system clock ÷ 6 110 TCK is system clock ÷ 7 111 TCK is system clock ÷ 8
30 jtagm_JCOMP	JTAG reset.

Table continues on the next page...

JTAGM_MCR field descriptions (continued)

Field	Description
	0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

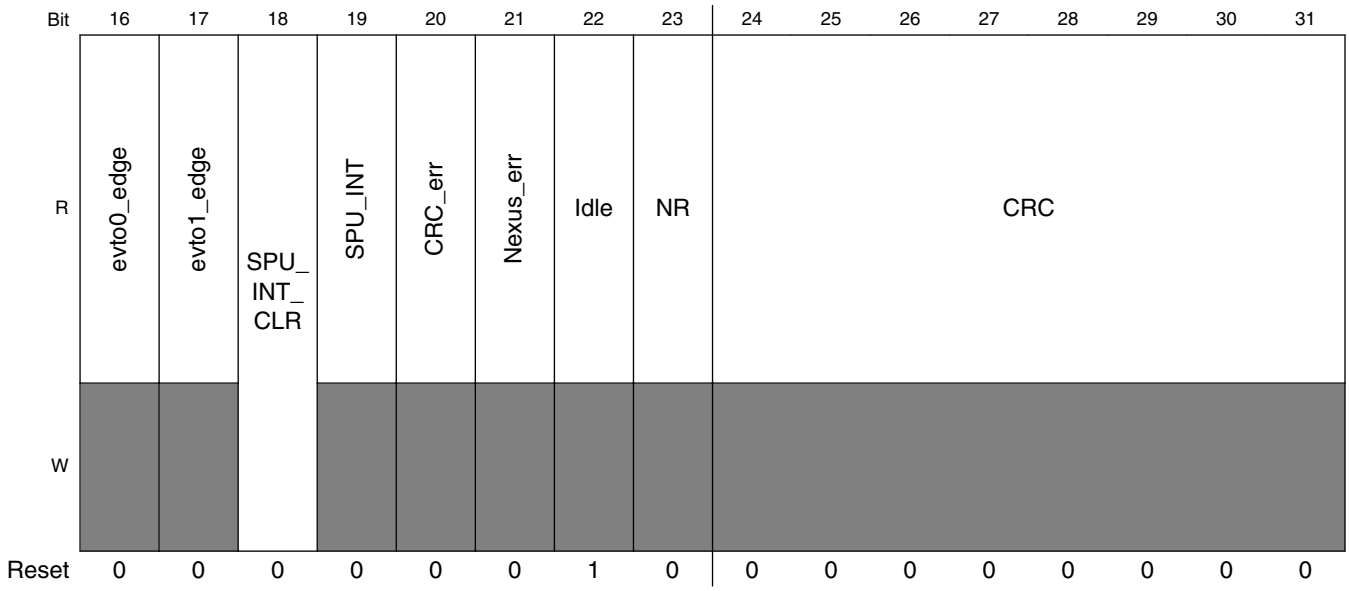
6.4.2 Status Register (JTAGM_SR)

The functionality of some of the SR bits changes depending on the setting of the DTM bit in the MCR. These differences are described in the field descriptions table.

In the ED, all bits of this register need to be accessed in the BD instance of the JTAGM. The exceptions are the bits related to the SPU interrupts (SPU_INT, SPU_INT_CLR), which always need to be accessed from the PD JTAGM, regardless of being ED or not.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Overrun	0	TXGOOD	TXERROR	RXOVFL	INVFPS	INVICLC	ILLLCT	Reserved	LVDSEN	JTAGEN	LVDSSAFE	JTAGSAFE	LVDSESC	LFASTEN	TOOL
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0



JTAGM_SR field descriptions

Field	Description
0 Overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 TXGOOD	Status information from the LFAST to be used by software: Indicates either data frame or ping frame was transmitted successfully
3 TXERROR	Status information from the LFAST to be used by software: Indicates Tx data Interface not enabled and a frame is ready to be transmitted
4 RXOVFL	Status information from the LFAST to be used by software: Indicates Rx data FIFO overflow
5 INVFPS	Status information from the LFAST to be used by software: Indicates reception of frame with invalid frame payload size
6 INVICLC	Status information from the LFAST to be used by software: Indicates reception of frame with invalid ICLC code
7 ILLLCT	Status information from the LFAST to be used by software: Indicates reception of illegal LCT frame
8 Reserved	This field is reserved. Reserved to 0
9 LVDSEN	DCI status: DCI LVDS port enable
10 JTAGEN	DCI status: DCI JTAG port enable
11 LVDSSAFE	DCI status: DCI LVDS safe mode
12 JTAGSAFE	DCI status: DCI JTAG safe mode
13 LVDSESC	DCI status: DCI LVDS escape mode

Table continues on the next page...

JTAGM_SR field descriptions (continued)

Field	Description
14 LFASTEN	DCI status: DCI enable LFAST
15 TOOL	DCI status: DCI tool present, set when a debug tool is connected
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit. 0 No activity sensed on $\overline{\text{EVTO}}$ 0 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO}}$ 0
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO}}$ 1 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO}}$ 1
18 SPU_INT_CLR	Write 1 clears the SPU_INT. This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared. The BD JTAGM instance does not receive signals from the SPU. If required, this bit needs to be read from the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit. The BD JTAGM instance does not receive signals from the SPU. If required, this bit needs to be read from the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to 1 when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. In LFAST mode (DTM = 0), this bit has no effect and should not be used. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read
24–31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode (DTM = 0): Bit 24:31 contains the CRC. Write has no effect. SW Mode (DTM = 1): • Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. • Writing 1 on bit 24 clears evto0_edge, bit 16 • Writing 1 on bit 25 clears evto1_edge, bit 17

6.4.3 Data Out Register 0 (JTAGM_DOR0)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TMS_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR0 field descriptions

Field	Description
0–31 TMS_HIGH	Higher word of data for TMS, bits 59-28. Writable by software only when DTM = 1

6.4.4 Data Out Register 1 (JTAGM_DOR1)

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TMS_LOW																												Reserved			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR1 field descriptions

Field	Description
0–27 TMS_LOW	Lower word of data for TMS, bits 27-0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first
28–31 Reserved	This field is reserved.

6.4.5 Data Out Register 2 (JTAGM_DOR2)

Address: 0h base + 10h offset = 10h

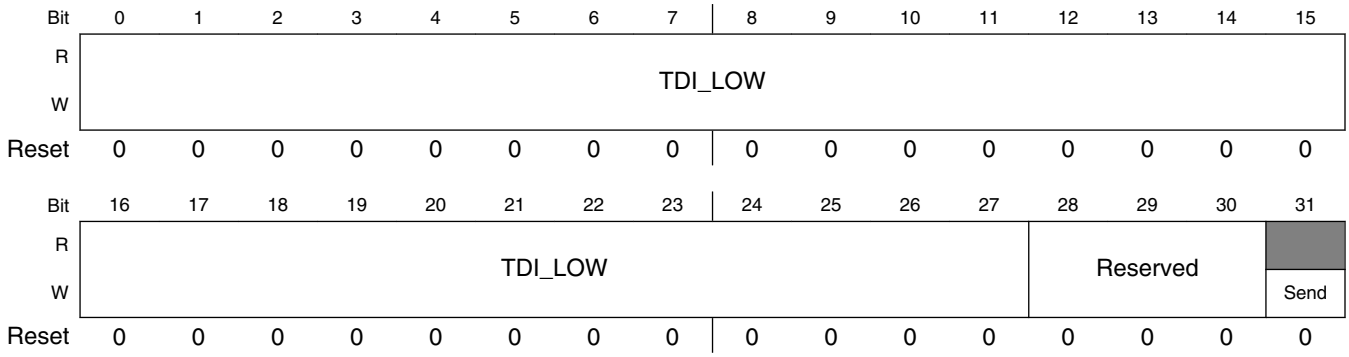
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDI_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR2 field descriptions

Field	Description
0–31 TDI_HIGH	Higher word of data for TDI, bits 59-28. Writable by software only when DTM = 1

6.4.6 Data Out Register 3 (JTAGM_DOR3)

Address: 0h base + 14h offset = 14h

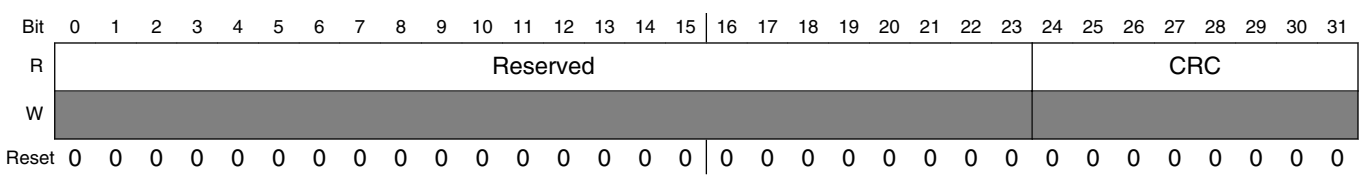


JTAGM_DOR3 field descriptions

Field	Description
0–27 TDI_LOW	Lower word of data for TDI, bits 27-0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
28–30 Reserved	This field is reserved.
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

6.4.7 Receive CRC Register (JTAGM_RxCRC)

Address: 0h base + 18h offset = 18h

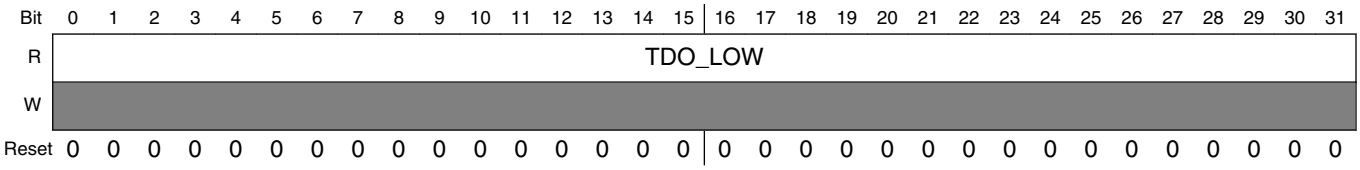


JTAGM_RxCRC field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

6.4.8 Data Input Register 0 (JTAGM_DIR0)

Address: 0h base + 1Ch offset = 1Ch

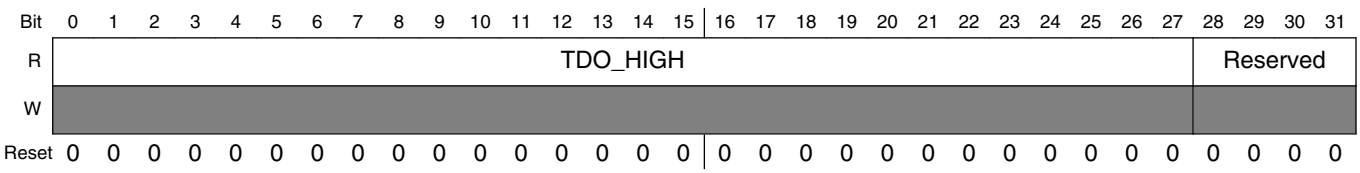


JTAGM_DIR0 field descriptions

Field	Description
0–31 TDO_LOW	Lower word of data received on TDO, bits 0-31

6.4.9 Data Input Register 1 (JTAGM_DIR1)

Address: 0h base + 20h offset = 20h



JTAGM_DIR1 field descriptions

Field	Description
0–27 TDO_HIGH	Higher word of data received on TDO, bits 32-59
28–31 Reserved	This field is reserved.

Chapter 7
Buddy Device—Nexus Aurora Router (NAR)

7.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in the following figure. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured. The Nexus Aurora Link (NAL) takes the 32-bit Nexus message beats from the NAR and processes them in Aurora-ready data octets for the Nexus Aurora Phy (NAP) block, which interfaces directly with the high-bandwidth port. The Aurora output port of NAR can be fed as a client to another NAR.

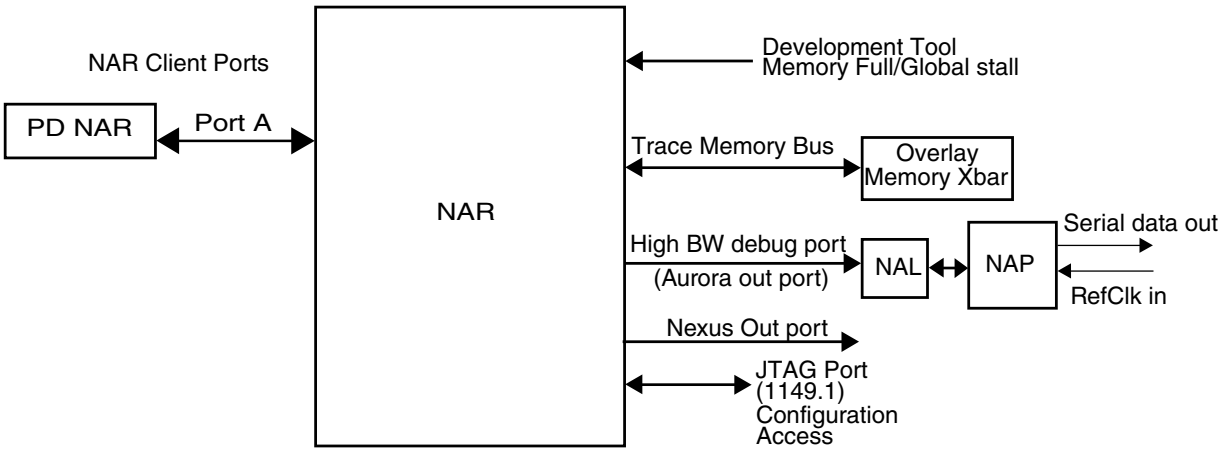


Figure 7-1. NAR top level block diagram

The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. The Nexus Out port provides additional flexibility to the channel trace data to memory on the Nexus recipient. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. These two partitions of the main queue are supported for better throughput each having 50% size. The two NAR output ports can be in operation simultaneously, with the output arbitration block splitting out messages by type or by source. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The Buddy Device (BD) NAR supports the following client interface:

- Aurora type interface with clock domain transfer at the same frequency

7.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (IEEE 1149.1) port.

7.3 Features

The NAR supports the following features:

- Connects 1 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Nexus out port for direct connection to pins (low-speed applications) or to another Nexus port controller
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Both output ports may operate simultaneously. Filtering logic sorts messages to determine destination port.
- Partitionable main queue makes it possible to simultaneously service multiple output ports. It can also be used to devote more bandwidth to high-priority message-types or split trace stream between on-chip memory and trace port.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client

- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Programmable Global stall signal on debug tool indication of memory overflow
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

7.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (IEEE 1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 7-1. NAR registers

Register name	Read/Write	JTAG index (NAR relative)
NAR_CR—NAR Control Register	R/W	0
NAR_ST—NAR Status Register	R	1
NAR_SFR—Source Filter Register	R/W	2
NAR_TFR—Type Filter Register	R/W	3
NAR_TBALO—Trace memory port base address, low	R/W	4
NAR_TBAHI—Trace memory port base address, high	R/W	5
NAR_TCR—Trace Memory bus control register	R/W	6
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)
NAR_CSSR–Client Suppress Status Register	R	B
NAR_CDR—Client Disable Register	R/W	C
NAR_NAPCR—Nexus Aurora Phy Control Register	R/W	D
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E

7.4.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following table shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	MSP		0	ALT		0	AFA	0	FSD	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CR is described in the following table.

Table 7-2. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See Disabled for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). **Note:** Some clients may only support timestamps on every message. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled

Table continues on the next page...

Table 7-2. NAR_CR field descriptions (continued)

Field	Description
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 1/2 of queue is dedicated to alternate partition 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.
10–9 MSP	Message Select Port. Defines which port is the target for message type/source filtering. Only active in two-port applications. 00 No message select port. 0–1 port is active. 01 HBDP is the MSP 10 Trace memory bus is the MSP 11 Nexus is the MSP
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 HBDP is the ALT 10 Trace memory bus is the ALT 11 Nexus is the ALT
5	Reserved
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2 FSD	Frame Serial Data. 0 Framing disabled. Data is streamed through the Aurora out port. 1 Framing enabled. Start/End frame symbol inserted between every 256 Aurora words.
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode

Table continues on the next page...

Table 7-2. NAR_CR field descriptions (continued)

Field	Description
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

7.4.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following table shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				MPF	MLV		APF	ALV		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The NAR_ST register is described in the following table.

Table 7-3. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode 001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode
27–24	NAR Config Error.

Table continues on the next page...

Table 7-3. NAR_ST field descriptions (continued)

Field	Description
NCE	See Table 7-18 for coding
23 MPF	MSP Partition Full. 0 MSP partition not full 1 MSP partition full
22–21 MLV	MSP Partition Queue Level. This field is available only when both output ports are enabled i.e. the main queue is partitioned. 00 0% < queue level < 25% 01 25% < queue level < 50% 10 50% < queue level < 75% 11 75% < queue level <100%
20 APF	ALT Partition Full. 0 ALT partition not full 1 ALT partition full
19–18 ALV	ALT Partition Queue Level. 00 0% < queue level < 25% 01 25% < queue level < 50% 10 50% < queue level < 75% 11 75% < queue level <100%
17–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer of ALT partition (last valid write + 1)

7.4.3 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See Output arbitration, for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

7.4.3.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFI	FE0	SF0				0	0	0	FE1	SF1				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	SF2				0	0	0	FE3	SF3				0	ACC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 7-4. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters

Table continues on the next page...

Table 7-4. NAR_SFR field descriptions (continued)

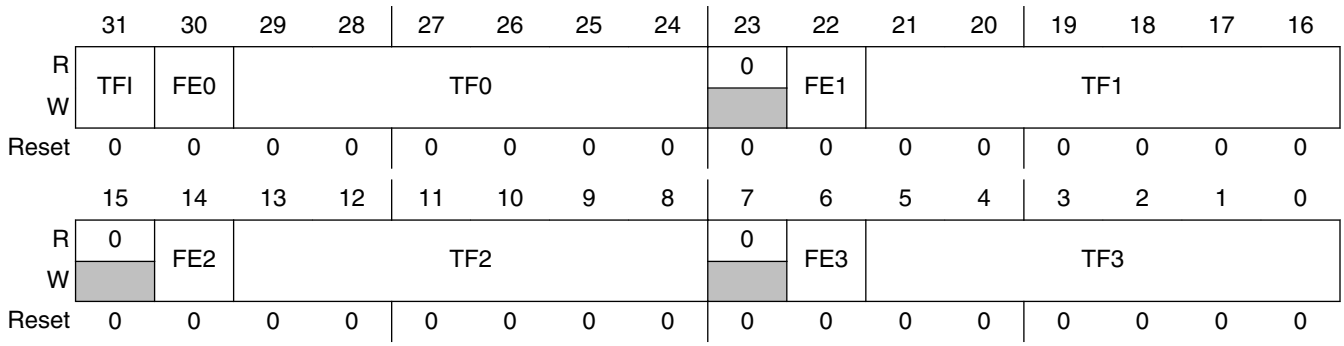
Field	Description
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See Output arbitration for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See Output arbitration for an example.
17–15	Reserved
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See Output arbitration for an example.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See Output arbitration for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

7.4.3.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

Table continues on the next page...



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TFR is described in the following table.

Table 7-5. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See Output arbitration for an example.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See Output arbitration for an example.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See Output arbitration for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See Output arbitration for an example.

7.4.4 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

7.4.4.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following tables show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBAHI																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register index: 4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBALO																						ADRLO[9]	ADRLO[8]	ADRLO[7]	ADRLO[6]	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 7-6. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 7-7. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

7.4.4.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following table shows the format of the NAR_TCR.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTM	BTS		BTH		0	0	0	TBW	MXFR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

R W	MXFR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TCR is described in the following table.

Table 7-8. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
23 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

7.4.5 Suppress mode triggers registers (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in Suppress mode. The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress

Trigger Control Register, shown in the following table, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7-A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LST															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 7-9. NAR_STCR field descriptions

Field	Description	
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode	
29–16	Reserved	
15–0 LST	Local Suppress Trigger.	
	0000000000000000	The associated trigger is disabled for local suppress mode for all clients
	xxxxxxxxxxxxxxx1	Trigger is enabled for Client 1 local suppress
	xxxxxxxxxxxxxx1x	Trigger is enabled for Client 2 local suppress
	xxxxxxxxxxxxx1xx	Trigger is enabled for Client 3 local suppress
	xxxxxxxxxxxx1xxx	Trigger is enabled for Client 4 local suppress
	xxxxxxxxxxx1xxxx	Trigger is enabled for Client 5 local suppress
	xxxxxxxxxx1xxxxx	Trigger is enabled for Client 6 local suppress
	xxxxxxxxx1xxxxxx	Trigger is enabled for Client 7 local suppress
	xxxxxxxx1xxxxxxx	Trigger is enabled for Client 8 local suppress
	xxxxxxx1xxxxxxxx	Trigger is enabled for Client 9 local suppress
	xxxxxx1xxxxxxxxx	Trigger is enabled for Client 10 local suppress
	xxxxx1xxxxxxxxxx	Trigger is enabled for Client 11 local suppress
	xxxx1xxxxxxxxxxx	Trigger is enabled for Client 12 local suppress
	xxx1xxxxxxxxxxxx	Trigger is enabled for Client 13 local suppress
	xx1xxxxxxxxxxxxx	Trigger is enabled for Client 14 local suppress
	x1xxxxxxxxxxxxxx	Trigger is enabled for Client 15 local suppress
	1xxxxxxxxxxxxxxx	Trigger is enabled for Client 16 local suppress
	All other options	Reserved

7.4.6 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following table, can then be read to determine which clients are suppressed, and which are active.

Register index: B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 7-10. NAR_CSSR field descriptions

Field	Description	
31–16	Local Suppress Status.	
LSS	0000000000000000	No client is in local suppress mode
	xxxxxxxxxxxxxxx1	Client 1 is suppressed
	xxxxxxxxxxxxxx1x	Client 2 is suppressed
	xxxxxxxxxxxxx1xx	Client 3 is suppressed
	xxxxxxxxxxxx1xxx	Client 4 is suppressed
	xxxxxxxxxxx1xxxx	Client 5 is suppressed
	xxxxxxxxxx1xxxxx	Client 6 is suppressed
	xxxxxxxxx1xxxxxx	Client 7 is suppressed
	xxxxxxxx1xxxxxxx	Client 8 is suppressed
	xxxxxxx1xxxxxxxx	Client 9 is suppressed
	xxxxxx1xxxxxxxxx	Client 10 is suppressed
	xxxxx1xxxxxxxxxx	Client 11 is suppressed
	xxxx1xxxxxxxxxxx	Client 12 is suppressed
	xxx1xxxxxxxxxxxx	Client 13 is suppressed
	xx1xxxxxxxxxxxxx	Client 14 is suppressed
	x1xxxxxxxxxxxxxx	Client 15 is suppressed

Table continues on the next page...

Table 7-10. NAR_CSSR field descriptions (continued)

Field	Description	
	1xxxxxxxxxxxxxx	Client 16 is suppressed
	All other options	Reserved
15–0	Reserved	

7.4.7 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following table, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Note

Even though the BD NAR is described as only having one client, there is actually a second dummy client, client 2. To re-allocate the dummy RX queue to client 1 (the PD NAR), set the BD NAR client 2 RQD bit.

Register index: C

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAS																RQD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 7-11. NAR_CDR field descriptions

Field	Description	
31–16	Client Activity Status.	
CAS	xxxxxxxxxxxxxxx1	Client 1 is active with new data
	xxxxxxxxxxxxxx1x	Client 2 is active with new data
	xxxxxxxxxxxxx1xx	Client 3 is active with new data
	xxxxxxxxxxxx1xxx	Client 4 is active with new data

Table continues on the next page...

Table 7-11. NAR_CDR field descriptions (continued)

Field	Description	
	xxxxxxxxxxx1xxxx	Client 5 is active with new data
	xxxxxxxxxx1xxxxx	Client 6 is active with new data
	xxxxxxxxx1xxxxxx	Client 7 is active with new data
	xxxxxxxx1xxxxxxx	Client 8 is active with new data
	xxxxxxx1xxxxxxxx	Client 9 is active with new data
	xxxxxx1xxxxxxxxx	Client 10 is active with new data
	xxxxx1xxxxxxxxxx	Client 11 is active with new data
	xxxx1xxxxxxxxxxx	Client 12 is active with new data
	xxx1xxxxxxxxxxxx	Client 13 is active with new data
	xx1xxxxxxxxxxxxx	Client 14 is active with new data
	x1xxxxxxxxxxxxxx	Client 15 is active with new data
	1xxxxxxxxxxxxxxx	Client 16 is active with new data
	All other options	Reserved
15–0	Receive Queue Disable.	
RQD	xxxxxxxxxxxxxxx1	Disable receive queue for Client 1
	xxxxxxxxxxxxxx1x	Disable receive queue for Client 2
	xxxxxxxxxxxxx1xx	Disable receive queue for Client 3
	xxxxxxxxxxxx1xxx	Disable receive queue for Client 4
	xxxxxxxxxxx1xxxx	Disable receive queue for Client 5
	xxxxxxxxxx1xxxxx	Disable receive queue for Client 6
	xxxxxxxxx1xxxxxx	Disable receive queue for Client 7
	xxxxxxxx1xxxxxxx	Disable receive queue for Client 8
	xxxxxxx1xxxxxxxx	Disable receive queue for Client 9
	xxxxxx1xxxxxxxxx	Disable receive queue for Client 10
	xxxxx1xxxxxxxxxx	Disable receive queue for Client 11
	xxxx1xxxxxxxxxxx	Disable receive queue for Client 12
	xxx1xxxxxxxxxxxx	Disable receive queue for Client 13
	xx1xxxxxxxxxxxxx	Disable receive queue for Client 14
	x1xxxxxxxxxxxxxx	Disable receive queue for Client 15
	1xxxxxxxxxxxxxxx	Disable receive queue for Client 16
	All other options	Reserved

7.4.8 Nexus Aurora Phy control register (NAR_NAPCR)

The Nexus Aurora Phy control register provides configuration control to number of HBDP lanes of Nexus Aurora Phy and LVDS pad control signals. The following table shows the format of the NAR_NAPCR.

Register index: D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	NUM_LANE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0		0	LVDS_TXOPT				LVDS_RX_TERM_EN	LVDS_BIAS_PRG	LVDS_RXOPT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The NAR_NAPCR is described in the following table.

Table 7-12. NAR_NAPCR field descriptions

Field	Description
19–16 NUM_LANE	TX Lane Configuration. Number of enabled TX lanes. This field is device specific. For a description of this field see the Calibration and Debug configuration chapter which describes how the modules are configured.
11–8 LVDS_TXOPT	This is an internal characterization field.
7 LVDS_RX_TERM_EN	This is an internal characterization field.
6 LVDS_BIAS_PRG	This is an internal characterization field.
5–0 LVDS_RXOPT	This is an internal characterization field.

7.4.9 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following table shows the format of the NAR_AHFPAR.

Register index: E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Register definition

	0	0	0	0	0	0	0		0	0	0	
R	0	0	0	0	0	0	0	GSE	0	0	0	GLOBAL_STALL_COUNT
W												
Reset	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	AHB65kEN	AHB01kEN	PART_CFG		AHB_65K_Marker				AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_AHFPAR is described in the following table.

Table 7-13. NAR_AHFPAR field description

FIELD	Description
31-25	Reserved and must be written with 0s
24 GSE	Global Stall Enable. Setting this bit enables the NAR to act on dci_evti[1] event to generate a global stall
23-21	Reserved and must be written with 0s
20-16 GLOBAL_STALL_COUNT	Global Stall Count. Value at this field determines how many cycles, in the NAR's operating clock, the global stall remains asserted on a dci_evti event. This is a five bit field so the maximum value is 31 cycles. See Global (Development tool client) stall control
15-12	Reserved and must be written with 0s
11 AHB65kEN	Enable 65K Marker Event
10 AHB01kEN	Enable 1K Marker Event
9-8 PART_CFG	Partition Configuration. 00 No partition 01 Reserved 10 50% partition (in two partition two configuration case) 11 Reserved
7–4 AHB_65K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 65K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Overlay and internal trace memory full status generation **Note:** This is used in BD as it has memory ranges from 65K to 983K
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Overlay and internal trace memory full status generation **Note:** This is used in PD as it has memory ranges from 1K to 15K

7.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

7.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/ filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note

The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see Suppress mode. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

7.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in the following table.

Table 7-14. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

7.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

7.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

7.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

7.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

7.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in the following table on the incoming messages before transmitting to one of the output ports.

Table 7-15. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8 × 32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8 × 32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

7.5.3 Data reception and arbitration

The BD NAR supports 1 client (although a second dummy client is used to increase the RX buffer size for client 1). Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There is one type of client supported at NAR client interface. One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits.

When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the

client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn × 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

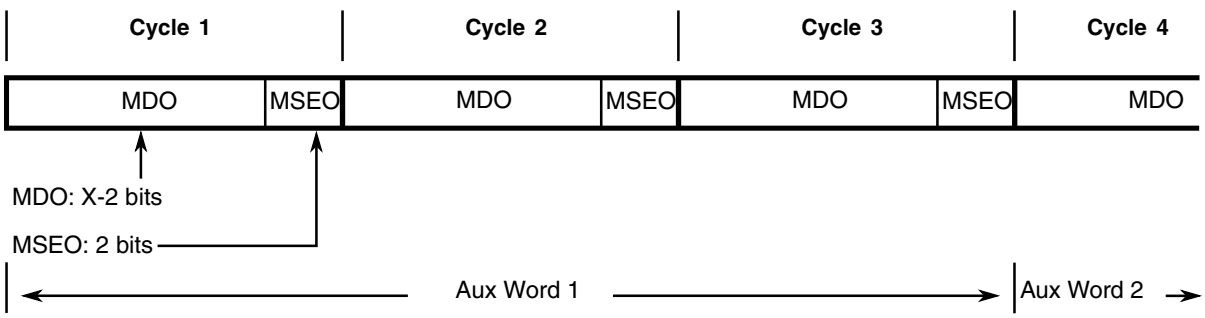


Figure 7-2. AUX bus parsing

Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

7.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR

can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

7.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the

message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

7.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type and/or messages from a particular client, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Even though the NAR supports three outputs (HBDP, Nexus, and trace memory bus) only two can be used at the same time. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register

(NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types and/or four sources.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0) or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is ~(data_trace) AND (source = 13)HBDP

Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

7.5.7 Queue partitioning

It is possible to partition the queue into two sections, one that supports the MSP, and the other that supports the alternate port as shown in Figure 7-3. Messages are routed, based on the MSP bit field, to either the MSP queue partition or the alternate queue partition. This allows both ports to be active simultaneously, since messages can be taken directly from both partitions. Each partition uses an accumulator to store a full central queue width beats of data before writing into memory, since on any given cycle the full Nexus word out of receiver arbitration may be split between the two partitions. Queue partitioning is enabled by setting the AQP bits of the NAR_CR.

The main advantage of using a partitioned queue is that it can improve data throughput and performance when multiple ports are active. It allows the trace memory bus port to utilize block transfer mode, which also improves performance. The only drawback is that queue utilization efficiency can be negatively impacted.

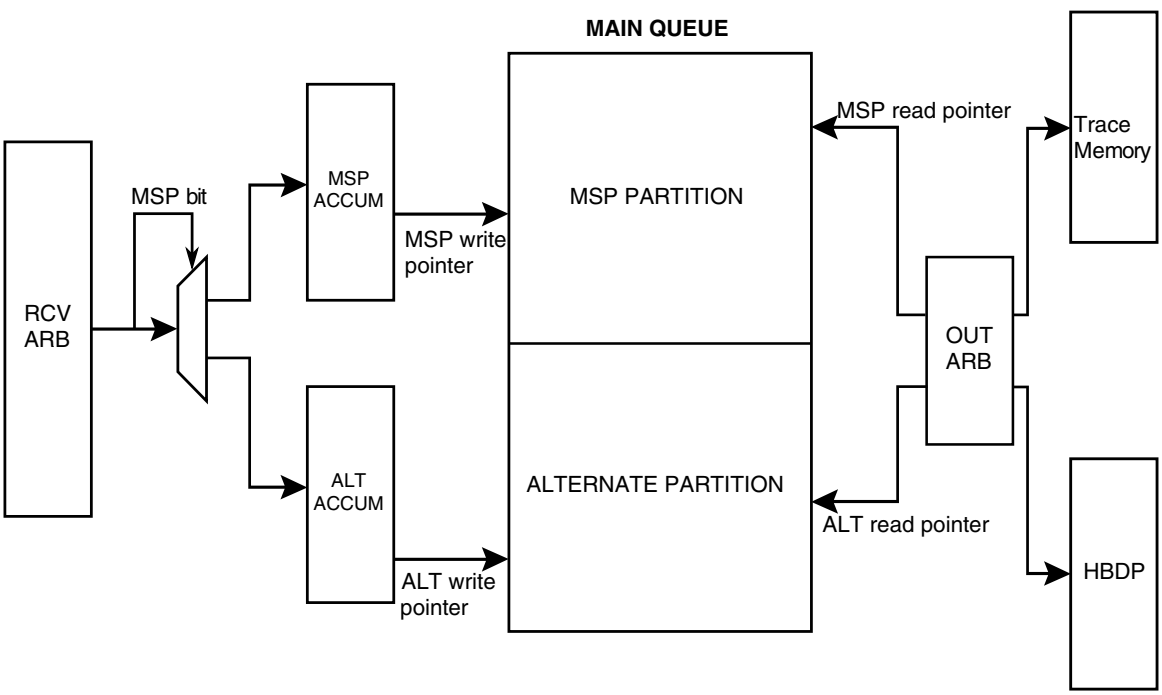


Figure 7-3. Queue partitioning

7.5.8 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

7.5.8.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16 × 4-bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4 × 16-bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = 16 × number of lanes).

7.5.8.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

7.5.8.3 Nexus format

If the Nexus port is used, entries are sent directly out of the queue with no translation or resizing. The Nexus output, Nexus input, and queue are all the same width.

The Nexus port also includes an EVTO output. This allows EVTO signaling functionality to be maintained in systems with multiple NARs. EVTO can also be used as a trigger by an external event processing block, to allow events such as client halt to occur based on

NAR events such as tracebuffer overflow. Currently, the only event that can cause an EVTO assertion is an HBDP hard error/reset, since data is almost certainly lost when the HBDP link goes down, the NAR needs to communicate that back to the clients so that they can generate sync messages.

Many Nexus message types use address/data compression, wherein the compressed data is generated by XORing the current data with the previous data sent. This type of compression can only work if there is no data lost; if data is lost, the client must start over by sending a new sync message which contains the full uncompressed address or data.

7.5.9 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

7.5.10 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 × 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

7.5.11 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.

- Watchpoint—The NAR generates its own watchpoint match message under the circumstances shown in Table 7-19.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 7-16. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message[1]	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC [2]	NAR source ID
	4	4	fixed	ETYPE	See Table 7-17
	12	12	fixed	ECODE	See Table 7-18
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message[3]	6	6	fixed	TCODE	TCODE value = 15
	6	6	fixed	SRC [2]	NAR source ID
	6	6	fixed	WPHIT	See Table 7-19
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2012 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2012 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

7.5.11.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in the following table.

Table 7-17. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see Table 7-18.
0011 (h'3)	Trace memory bus port read/ write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.[2]
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See Table 7-18	The specific configuration error is conveyed in bits 0–3 of the ECODE field (Table 7-18), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1101 (h'D)	Receive Queue Overrun	Minor	Drop message	If a client sends a data word that the NPC cannot accept (due to a full receive queue), the new data word will be dropped2.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.

Table continues on the next page...

Table 7-17. NAR error types (continued)

ETYPE	Description	Classification	NAR action	Comment
1111 (h'F)	Stall error	Minor	—	—

2. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in the following table.

Table 7-18. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

7.5.11.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

Table 7-19. Watchpoint hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

7.5.12 Timestamp function

The NAR and its clients may transmit timestamps to allow precise ordering of events. Synchronous or asynchronous clients use the NAR timer value broadcasted over a bus to ensure that all timestamps are in sync as shown in the following figure. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

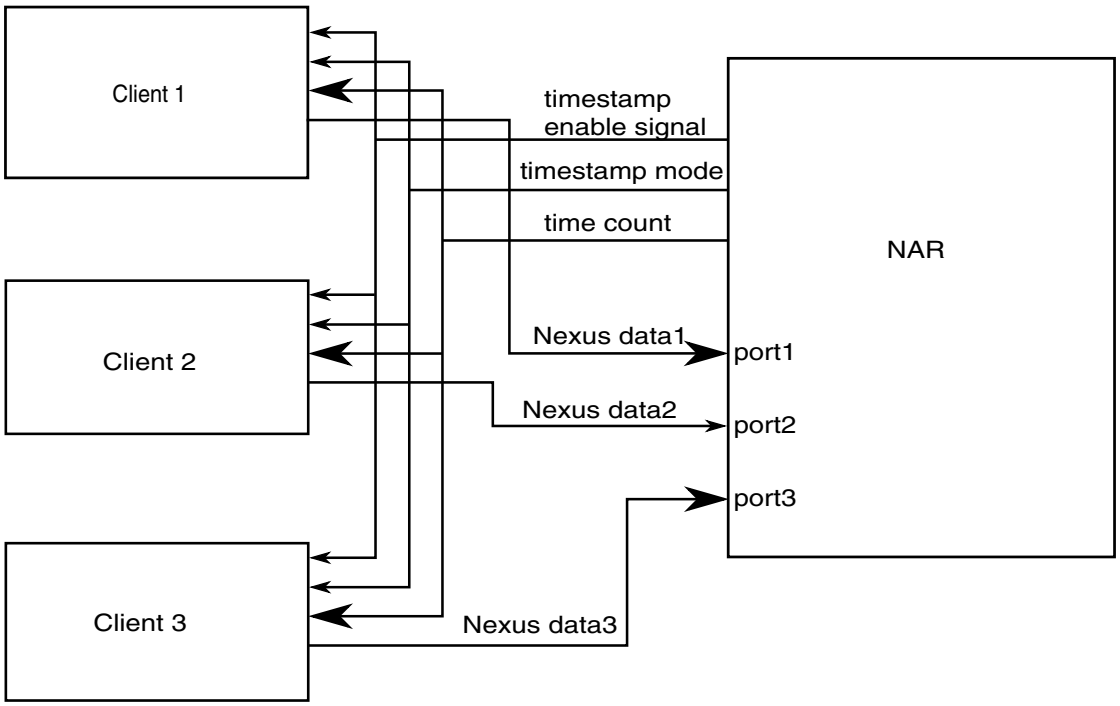


Figure 7-4. Timestamp architecture

Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The timestamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode, and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

7.5.13 Global (Development tool client) stall control

The external development tool has finite memory. To prevent data loss when this memory is full, the tool gives an event to the device. The DCI block processes this and forwards to NAR block. The dci_evti[1] carries this information. NAR detects the falling edge of dci_evti[1] and loads a counter with a configurable value programmed in GLOBAL_STALL_COUNT field of NAR_AHFPAR. This counter starts decrementing immediately, reaches to zero and stops. During the running period of the counter the Global Stall signal remains asserted thus indicating to stall the message interface to the NAL by that many cycles. The GLOBAL_STALL_COUNT field is five bits so client can be stalled up to 31 cycles.

7.5.14 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
  - AHB start (base) address—Base address = 0x0D00_0000 (i.e.TBAHI = 0x0, TBALO = 0x0D00_0000
  - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
  - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is (0xC0 × 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

Similarly for 65 KB boundary matching NAR_AHFPAR[7:4] needs to be programmed and the corresponding enable bit needs to be set.

7.5.15 Configuration/debug interface JTAG

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in Table 7-20. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

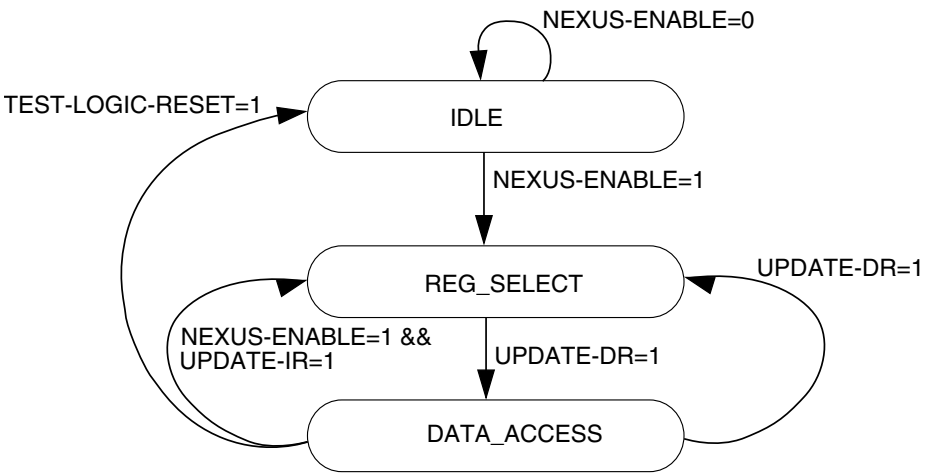


Figure 7-5. Nexus controller state machine

Table 7-20. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

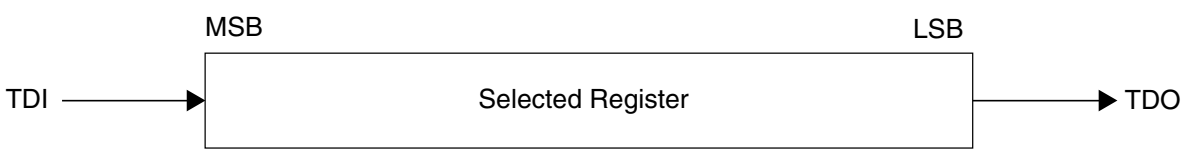


Figure 7-6. Shifting data into register

7.5.15.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

7.5.15.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 7-5, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 7-21. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

7.5.15.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following figure. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in Table 7-1.

Table 7-22. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state.When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state.When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 7-23. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

7.5.16 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
  - Early Write (for higher BW in special module)
  - 32-bit address bus width
  - 64-bit data bus width
  - Transfer Types
  - IDLE, NONSEQ, SEQ
  - Single Transfers
  - Burst Transfers
  - Configurable support for INCR
  - Configurable support for INCR4, INCR8, INCR16, WRAP4
  - Response Type
  - OKAY, ERROR
  - Transfer Size
  - Byte, Half Word, Word, Double-Word
  - Protection Control
  - Cacheable/ Non Cacheable Access Control
  - User/Privilege Access
- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

7.5.16.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

7.5.16.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

7.5.16.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

7.5.17 Aurora client controller

This block converts the Aurora client protocol from the PD NAR into an internal client protocol that is accepted by the NPC block. The Nexus beat size must be same at both interfaces. The relation between the Aurora client data width and the NPC client interface data width must be maintained as described here.

The Aurora data width is defined as the Aurora width × 16 bits, where 16 bits comprise one Aurora word. The Nexus client data width is defined as Bn × 32, where 32 = MDO bits + 2 MSEO bits and Bn is the number of beats.

Thus:

Aurora data width = the Nexus client data

Aurora width × 16 = Bn × 32

Aurora width = 2 × Bn

This block works in a similar way to the legacy client controller block except for the interface level control. The Aurora client has the information about the FIFO size. It sends as many data as the FIFO size then waits for the data accept signal from this block. The controller gives the data accept only when a read occurs and sends data to the NPC interface. The FIFO act as a storage and serves the purpose of clock domain transfer as well.

7.6 Application information

The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in Output arbitration.

Example flow of NAR configuration for different cases.

Configuration with Two output port enabled

1. For Two port enabled one port must be Message Select (MSP) and other Alternate (ALT); Trace Memory and Aurora ports with multiple clients

   Selection of MSP port depends on user choice. One option could be based on the requirement of Filtering. If Messages from some known sources with some specific type (if any) is required to be sent to a particular port then select user can select that

port as MSP. All other messages will go to the Alternate port. The filter can be used as inverted manner where unmatched messages will go to MSP. (Please see section “Output arbitration” for details)

- Configure the Source Filter register (NAR_SFR), To enable proper filtering of messages based on Message source going to MSP or ALT port
  - Example: for DWPU connected in Nexus out port set the source ID of the core and NXMC in any two source filter type and enable it
- Configure the Type Filter register (NAR_TFR), To enable proper filtering of messages based on Message Type going to MSP or ALT port
  - Example: for DWPU connected in Nexus out port set the Type of messages in any two Type filter type and enable it
- Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
  - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
  - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
- Configure Client disable register NAR_CDR (disable clients which are not required)
- Configure NPC suppress trigger registers NAR_STCR[x] (x=0-3) (suppress the clients for which data is not to be forwarded at output) (Difference between suppressing and disabling is that suppressing can be done dynamically during NAR operation but disabling can’t be done this way which may leads to system hang)
- Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
- Configure the NAR control register for other controls and Enable NAR
  - Set the MSP as Nexus Out and ALT as Trace memory
  - Set AQP for partitioning the Main Queue for ALT and MSP ports
  - Set time stamp controls
  - Enable NAR to start operation
  - Configuration for Single Output port with single partition in NAR central Message Queue

2. Only Trace Memory port is enabled with multiple client
   - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
     - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
     - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
   - Configure Client disable register NAR_CDR (disable clients which are not required)

- Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
- Configure the NAR control register for other controls and Enable NAR
  - Set the Trace Memory port as ALT port (Alternate port is only the main port in single out port application)
  - Set AQP for disabling partitioning
  - Set time stamp controls
  - Set watch point controls
  - Enable NAR to start operation

Chapter 8
Buddy Device—Nexus Aurora Link (NAL)

8.1 Introduction

The Aurora Protocol is used to send debug information over a high-speed serial link out of the device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer SerDes and transceivers; it includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol. The following figure details the NAL block diagram. The number of lanes, n, is .

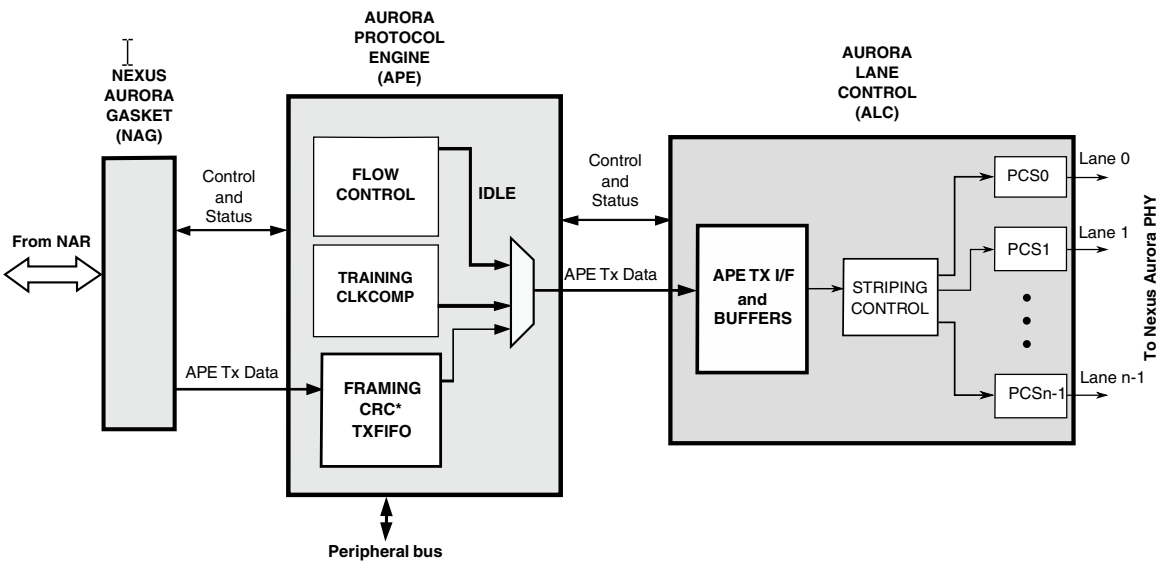


Figure 8-1. NAL block diagram

Debug data in the Nexus format from multiple clients is collected by the Nexus Aurora Router (NAR) ; messages are then packetized and sent across an asynchronous clock boundary-crossing gasket to the Aurora Protocol Engine, which performs all the logical layer functions and flow control; data from the protocol engine is sent on to the Aurora Lane Control (ALC) block, which performs data-striping and sizing, based on the number of lanes selected for debug; it also performs the PCS functions of the link. The PCS

drives the Nexus Aurora Physical (NAP) interface. The NAP, in conjunction with the NAL, supports one-way simplex high-speed serial communication with an external debugger.

The NAL is composed of three main blocks: the NAR/Aurora Gasket, the Aurora Protocol Engine, and Aurora Lane Control. The table below lists NAL functionality by block and lists the high level features for the Nexus Aurora PHY.

Table 8-1. NAL and NAP feature summary

Sub-Component	Feature
NAR Aurora Gasket (NAG)	• Transmit FIFO to buffer data as it crosses the NAR/NAL clock-domain boundary and Transmit FIFO space available indication to NAR. The NAR uses Transmit FIFO space available indication(s) to determine if data can be sent to the NAL. • Platform to Domain clock domain crossing. The NAR is on the core clock domain while the NAP is on a domain sourced from the external PHY clock input signals.
Aurora Protocol Engine (APE)	• Link training • Per the Aurora Protocol specification, clock-compensation, and data framing • Cyclic Redundancy Check (CRC) generator
Aurora Link Control (ALC)	• Striping of data stream from the Aurora core across all available lanes in two-byte chunks • PCS functionality for NAL
Nexus Aurora PHY (NAP)	• Physical transmission of serial data to device pins • External clock source for SerDes clock domain

8.2 Transmit operation

During transmit operation, the NAR collects debug data from Nexus clients over a 32-bit interface: 30 bits are used for MDO and 2 bits are used for MSEO information. The NAR places MDO data into a payload (maximum payload size depends on the number of lanes, each lane has a maximum of 7424 symbols before terminating the payload) and adds the following:

- 16-bit start symbol
- 16-bit end symbol
- 16-bit CRC if enabled before end of previous symbol

Each payload is transferred from the NAR to the NAR/NAL gasket, through the Aurora Protocol Engine, the Aurora Lane Control and finally out to the device SerDes/LVDS pins through the NAP.

After each end symbol packet transfer, the NAL adds a clock compensation sequence needed for downstream SerDes devices to remain synchronized to the MCU.

NOTE

Symbol counting for clock compensation starts just prior to the verification symbols; therefore, training symbols are excluded when counting.

8.3 Nexus Aurora formatting

The Aurora physical interface is the transport mechanism for Nexus messages. The Nexus messages are formatted by the Aurora formatter and striped into Aurora lanes. The Nexus messages themselves encapsulated into a virtual Nexus port that consists of a 30-bit Message Data Output (MDO) port and a 2-bit Message Start/End Output (MSEO) signal. The MSEO portion of the port is the two least significant bits (LSB) of the 32-bit Aurora message. The rest of the 32-bit Aurora frame consists of 30 MDO bits that are added LSB first.

8.4 Static training

The NAL supports static training. The NAL performs Static training by using programmable timers to determine how long to stay in each stage of the training process. In static training, there is no communication between the channel partners to exchange training status; rather the NAL sends the align/bond/verify sequence to its partner for a set amount of time sufficient for the receive partner to complete the training stage.

Note

Prior to initialization, the NAL must be reset to its POR state; to do this, assert the block reset. See the RST bit field description in the NAL General Control Register (NAL_GCR).

The following steps enable static training:

1. Program the counters in the NAL_TCR and set the STE bit
2. Set the RST bit

This resets the channel and start the training routine. The time spent in each stage can range from 16 to 3072 cycles; once a counter times out, a counter time-out signal is asserted to the Aurora Protocol Engine (APE) to prompt it to move to the next stage. Additionally, it is possible to hold the training procedure in any of the stages indefinitely, by setting one of the three hold bits (AHD/BHD/VHD). When set, these bits cause the

channel to sit in Align/Bond/Verify until unset, allowing a user to interactively step through the training procedure by monitoring the status at the debugger side and then stepping the device side to the next stage.

During the Aurora training sequence, the NAL generates a specific set of character sequences per the Aurora protocol specification which are to be transmitted simultaneously on all lanes.

8.5 Programmable registers

The Nexus Aurora Link (NAL) module provides a Nexus parallel to Aurora formatting for the MCU trace information. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 8-2. NAL memory map

Offset	Register name	Width (in bits)	Access	Reset value
0h	NAL General Status Register (NAL_GSR)	32	R	0008_0000h
8h	NAL General Control Register (NAL_GCR)	32	R/W	0000_0000h
Ch	NAL Training Control Register (NAL_TCR)	32	R/W	0000_0000h

8.5.1 Nexus Aurora Link General Status Register (NAL_GSR)

The NAL_GSR is a read only register that describes the status of the NAL.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												TXONLY	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			TXCFG			0									AS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-3. NAL_GSR field descriptions

Field	Description
31–20 Reserved	Reserved This bitfield is reserved.
19	Transmit Only mode. Indicates that Aurora is configured in a transmit-only mode.

Table continues on the next page...

Table 8-3. NAL_GSR field descriptions (continued)

Field	Description
TXONLY	0 Reserved 1 Aurora is configured for transmit-only simplex communication. Training method is configured to use Static Training.
18–13 Reserved	Reserved This bitfield is reserved.
12–10 TXCFG	TX Lane Configuration. Number of enabled TX lanes. These bits are read-only and are controlled external to the NAL. 000 Reserved 001 2 TX lanes 010 Reserved 011 4 TX lanes 100 Reserved 101 Reserved 110 Reserved 111 Reserved
9–1 Reserved	Reserved This bitfield is reserved.
0 AS	Aurora Status. 0 Aurora is not enabled. 1 Aurora is enabled.

8.5.2 NAL General Control Register (NAL_GCR)

The NAL_GCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RST	ALCPD	0												FCM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PCRST	0	SLD			0	BOOE	BOOD				CRCEB	CCOEN	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-4. NAL_GCR field descriptions

Field	Description
31 RST	Reset the Aurora Channel. 0 to 1 transition resets the channel and begins the training sequence. 0 normal operation 1 transition from 0 to 1 resets channel and starts training sequence
30 ALCPD	ALC/NAL Power Down. Gate off clocks to the NAL blocks (ALC and APE). 0 NAL is enabled with all clocks running provided other mechanisms of powerdown (such as alc_num_lanes) are not in effect. 1 NAL is powered down and inactive.
29–18 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.
17 FCM	Flow Control Mode. 0 Completion Mode 1 Immediate Mode
16-15 Reserved	Reserved and must be written with zero.
14 PCRST	Protocol Converter Reset. Resets the ALC when asserted.
13 Reserved	Reserved This read-only bit is reserved and always has the value zero.
12–10 SLD	Symbol Lock Delay. 000 Allow 3 characters to accumulate in elastic buffer before starting to read 001 Allow 4 characters to accumulate in elastic buffer before starting to read 010 Allow 5 characters to accumulate in elastic buffer before starting to read 011 Allow 6 characters to accumulate in elastic buffer before starting to read 100 Allow 7 characters to accumulate in elastic buffer before starting to read 101 Allow 8 characters to accumulate in elastic buffer before starting to read 110 Allow 9 characters to accumulate in elastic buffer before starting to read 111 Allow 10 characters to accumulate in elastic buffer before starting to read
9 Reserved	Reserved This read-only bit is reserved and always has the value zero.
8 BOOE	Bond Offset Override Enable. 0 ALC channel bond attempts to determine channel bond offset. 1 ALC uses user override setting for channel bond offset.
7–4 BOOD	Bond offset override data. Only valid when BOOE is set. 0000/1000 use 0-cycle offset 0001 use -7 cycle offset 0010 use -6 cycle offset 0011 use -5 cycle offset

Table continues on the next page...

Table 8-4. NAL_GCR field descriptions (continued)

Field	Description
	0100 use -4 cycle offset 0101 use -3 cycle offset 0110 use -2 cycle offset 0111 use -1 cycle offset 1001 use +1 cycle offset 1010 use +2 cycle offset 1011 use +3 cycle offset 1100 use +4 cycle offset 1101 use +5 cycle offset 1110 use +6 cycle offset 1111 use +7 cycle offset
3 CRCEB	CRC Enable. 0 CRC generation and checking is disabled for TX and RX data. 1 CRC is inserted for TX frames and CRC is checked for RX paths.
2 CCOEN	Clock Compensation Override Enable. This is a debug feature which allows the duration between clock compensation sequences to be reduced to 1280 cycles. 0 Clock compensation counter times out at 7424 cycles 1 Clock compensation counter times out at 1280 cycles
1–0 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.

8.5.3 NAL Training Control Register (NAL_TCR)

The NAL_TCR supports static training as described in Static training.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STE	AHD	BHD	VHD	Reserved					ATC				Reserved					BTC				Reserved						VTC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-5. NAL_TCR field descriptions

Field	Description
31 STE	Static Training Enable. 0 Reserved 1 Timer-based training method is used, ie. RX data is ignored and training is established based on ATC/BTC/VTC timers.

Table continues on the next page...

Table 8-5. NAL_TCR field descriptions (continued)

Field	Description
30 AHD	Hold in Align. 0 Follow Aurora training sequence 1 Remain in align until this bit is cleared (applies even during duplex-mode training)
29 BHD	Hold in Bond. 0 Follow Aurora training sequence 1 Remain in bond until this bit is cleared. (applies even during duplex-mode training)
28 VHD	Hold in Verify. 0 Follow Aurora training sequence 1 Remain in verify until this bit is cleared. (applies even during duplex-mode training)
27–23 Reserved	Reserved This bitfield is reserved and must be written with zero.
22–19 ATC	Align Timer Count 0000 Remain in align for 16 cycles 0001 Remain in align for 24 cycles 0010 Remain in align for 32 cycles 0011 Remain in align for 48 cycles 0100 Remain in align for 64 cycles 0101 Remain in align for 96 cycles 0110 Remain in align for 128 cycles 0111 Remain in align for 192 cycles 1000 Remain in align for 256 cycles 1001 Remain in align for 384 cycles 1010 Remain in align for 512 cycles 1011 Remain in align for 768 cycles 1100 Remain in align for 1024 cycles 1101 Remain in align for 1536 cycles 1110 Remain in align for 2048 cycles 1111 Remain in align for 3072 cycles
18–14 Reserved	Reserved This bitfield is reserved.
13–10 BTC	Bond Timer Count 0000 Remain in bond for 16 cycles 0001 Remain in bond for 24 cycles 0010 Remain in bond for 32 cycles 0011 Remain in bond for 48 cycles 0100 Remain in bond for 64 cycles 0101 Remain in bond for 96 cycles 0110 Remain in bond for 128 cycles

Table continues on the next page...

Table 8-5. NAL_TCR field descriptions (continued)

Field	Description
	0111 Remain in bond for 192 cycles 1000 Remain in bond for 256 cycles 1001 Remain in bond for 384 cycles 1010 Remain in bond for 512 cycles 1011 Remain in bond for 768 cycles 1100 Remain in bond for 1024 cycles 1101 Remain in bond for 1536 cycles 1110 Remain in bond for 2048 cycles 1111 Remain in bond for 3072 cycles
9–4 Reserved	Reserved This bitfield is reserved.
3–0 VTC	Verify Timer Count 0000 Remain in verify for 16 cycles 0001 Remain in verify for 24 cycles 0010 Remain in verify for 32 cycles 0011 Remain in verify for 48 cycles 0100 Remain in verify for 64 cycles 0101 Remain in verify for 96 cycles 0110 Remain in verify for 128 cycles 0111 Remain in verify for 192 cycles 1000 Remain in verify for 256 cycles 1001 Remain in verify for 384 cycles 1010 Remain in verify for 512 cycles 1011 Remain in verify for 768 cycles 1100 Remain in verify for 1024 cycles 1101 Remain in verify for 1536 cycles 1110 Remain in verify for 2048 cycles 1111 Remain in verify for 3072 cycles

Chapter 9
Buddy Device—Nexus Aurora PHY (NAP)

9.1 Introduction

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. Configuration settings in the Nexus Aurora Router (NAR) Nexus Aurora Phy control register (NAR_NAPCR) specify the NAP and NAL lane configuration. The NAL performs all logical layer functions, flow control, data-striping and sizing based on the number of data lanes available for use, and physical coding sublayer functions on the data being sent to the NAP for transmission. The NAP receives 8b/10b encoded data from the NAL then serializes and transmits this data through a given PHY lane's Low Voltage Differential Signaling (LVDS) buffers to an external debugger.

Each NAP lane consists of a data symbol character accumulator which registers the 10-bit character for that lane from the NAL on the rising edge of the NAL clock. Next, a serializer takes the 10-bit character and serializes it into the individual data bits that are then transmitted at the PHY clock rate via the LVDS output drivers to the external debugger receive channel.

Note

The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking in the received data stream.

The following figure shows a high level block diagram of the High Speed Nexus Interface and the NAP.

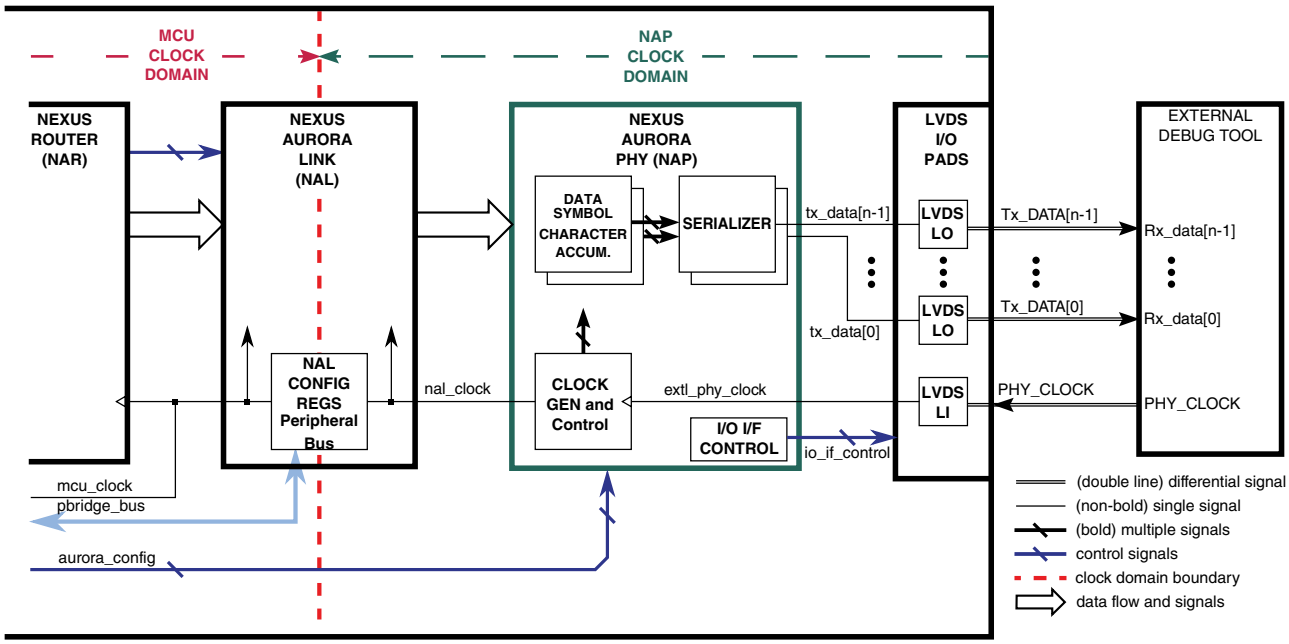


Figure 9-1. High speed Nexus interface block diagram

9.1.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.x compliant
- 1Gb/s per channel trace data payload bandwidth with 1.25 GHz clock
- Simplex mode with static (timer-based) channel training

9.1.2 Modes of operation

The operating mode of the NAP is determined by the device control signals. The functional configuration is determined by the NAR_NAPCR setting; functional operation is enabled by the NAR.

9.1.2.1 Reset

During Nexus Reset all internal NAP resources are reset, all output ports are disabled, and all inputs are ignored. Since there is no device clock domain logic in the NAP itself, the NAP is unaffected by device system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

9.1.2.2 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to ensure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

9.1.2.3 Transmit

During Nexus trace output operation, it is the responsibility of the NAR to push data into the Aurora Link at a sufficiently high rate to ensure that the link does not become data-starved. The NAL and NAP support up to 4 transmit lanes. The Aurora encapsulated data is modified by the NAL to ensure the appropriate number of symbols are transmitted through the NAP.

9.2 External signal description

The external signals of the NAP are listed in the following table. Each of the NAP signals consists of a positive and negative differential pair.

Table 9-1. NAP signals

Signal Name	Direction	Description
PHY_CLOCK (CLKP/CLKN)	Input	NAP clock input from external debug tool

Table continues on the next page...

Table 9-1. NAP signals (continued)

Signal Name	Direction	Description
TX Data0 (TX0P/TX0N)	Output	NAP Tx data for lane 0
TX Data1 (TX1P/TX1N)	Output	NAP Tx data for lane 1
TX Data2 (TX2P/TX2N)	Output	NAP Tx data for lane 2
TX Data3 (TX3P/TX3N)	Output	NAP Tx data for lane 3

9.3 Memory map and register definition

There are no control, configuration or status registers within the NAP. All control, configuration and status options for number of lanes and LVDS pads are handled in the NAR_NAPCR which is described in the NAR Chapter.

9.4 Functional description

High-level descriptions of the operation of the NAP and its main functional blocks are given in the following sections.

9.4.1 Data symbol character accumulator

The data symbol character accumulator portion of the NAP for each lane is made up of a symbol character register whose inputs are connected to the 10-bit NAL character bus. The data symbol character accumulator essentially registers a 10-bit character for that lane simultaneous with the rising edge of the NAL clock and presents the 10-bit quantity to the serializer at the appropriate times.

9.4.2 Serializer

The serializer portion of the NAP for each lane takes the data captured by that lane's Data Symbol Character Accumulator and demuxes the data to provide a single serial stream of symbol data to be output by the LVDS LO buffer for that lane at the PHY clock rate.

The following figure shows a high level timing diagram that demonstrates the operation of the serializer for a NAP lane, where S0–S9 represents a 10-bit symbol, S10–S19 represent the next symbol, and so on.

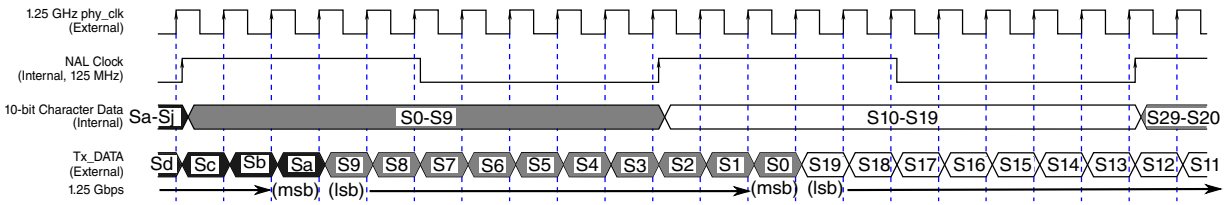


Figure 9-2. Serializer timing diagram

9.4.3 Clock generation and control

The Clock Generation and Control block is responsible for generating the necessary clocks and state signals needed by the rest of the NAP. The Data Symbol Character Accumulators and serializers are the main consumers of the signals generated by the Clock Generation and Control block.

9.5 Initialization information

The NAP does not require initialization other than the Nexus Reset and programming of the NAR_NAPCR to configure the number of lanes and LVDS pads.

Chapter 10
Buddy Device—Data Write Processing Unit (DWPU)

10.1 Introduction

The Data Write Processing Unit (DWPU) supports two specific debug and calibration functions:

- Data trace filtering
- Out of range parameter identification

DWPU takes the Nexus standard messages and processes the data write trace messages to filter out messages based on the tagged addressing schema contained in the Tag RAM. DWPU also generates triggers indicating out of range data values for a set of identified addresses and data ranges.

This figure shows the block diagram of the DWPU identifying various functional block and the data flow.

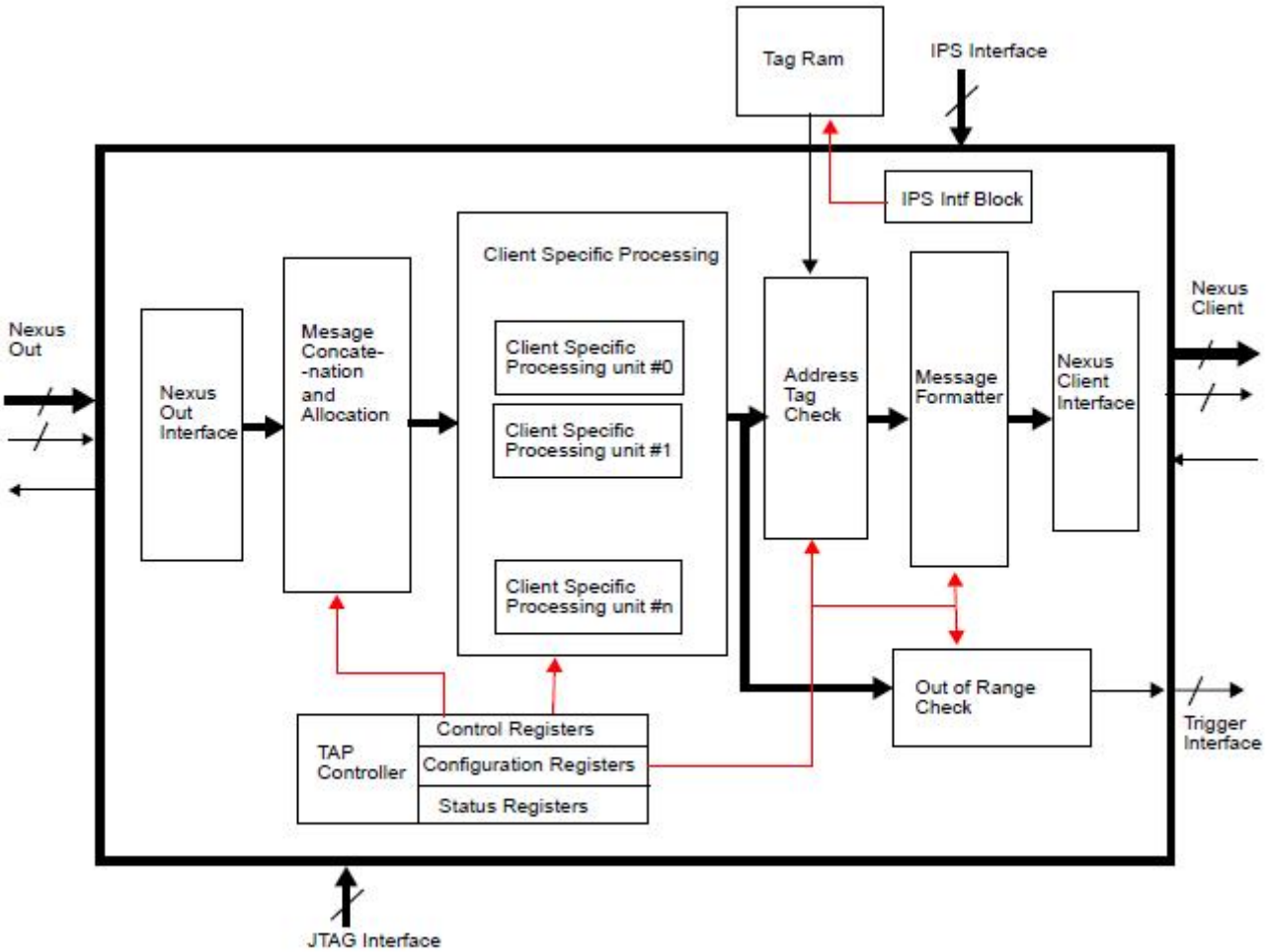


Figure 10-1. DWPU block diagram

10.2 Features

DWPU supports the following features:

- Out of range data value detection and data write trace filtering simultaneously from up to five Nexus clients
- Filtering operation that can be individually enabled or disabled for each 64-bit word in system RAM or data memory
- High bandwidth throughput
- Filtered trace stream is returned to the BD NAR to allow routing to the trace port or overlay RAM
- Detection of out of range data values writes within data write trace stream

10.3 Functional description

DWPU includes the following functional blocks:

- Nexus Out Interface (NOI)
- Message Concatenation and Allocation (MCA)
- Client Specific Processing (CSP)
- Address Tag Check (ATC)
- Out of Range Check (ORC)
- Message Formatter (MF)
- Nexus Client Interface (NCI)

10.3.1 Nexus Out Interface (NOI)

The Nexus Out Interface (NOI) enables DWPU to receive Nexus message for data write filtering from NAR. NOI maintains a minimum queue of two buffers to accommodate incoming messages. The NAR only initiates a transfer to the DWPU when it has all the beats of a messages. The incoming messages are buffered and then forwarded to the Message Concatenation and Allocation (MCA) for message concatenation and allocation. NOI ensures the temporal order of the incoming messages while forwarding to MCA. NOI also acknowledges the processing of data write messages by MCA.

The format for incoming message with NEXUS_WORD = 32 and NUM_BEAT_IN = 8 is shown below.

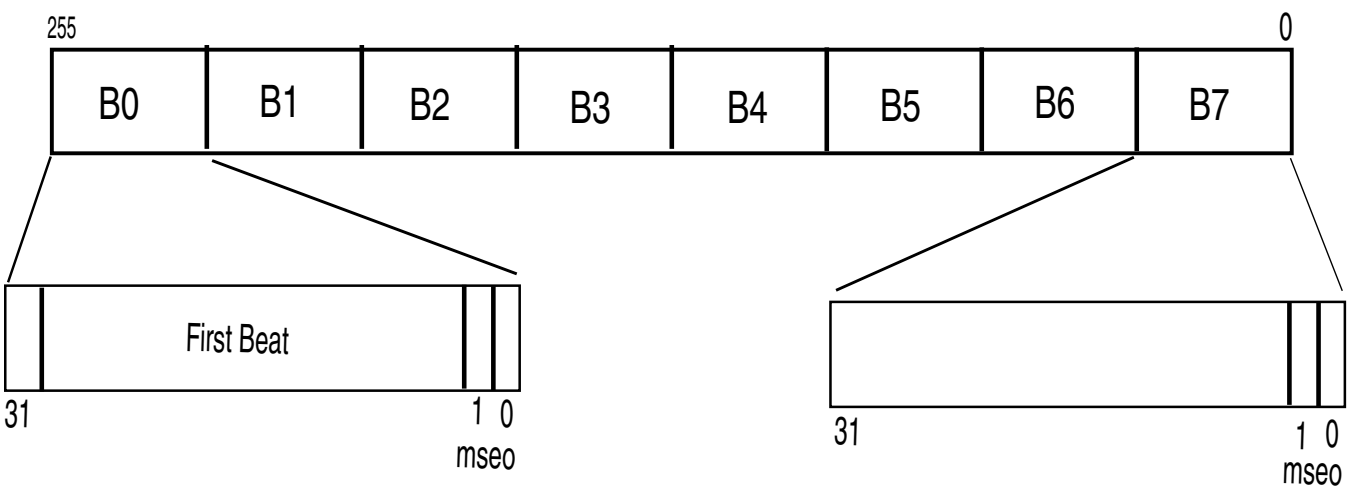


Figure 10-2. Incoming message format

10.3.2 Message Concatenation and Allocation (MCA)

Message Concatenation and Allocation (MCA) starts concatenating message beats once the NOI has a valid input messages to form a complete message. Concatenated messages with selected source IDs contained in within Nexus messages are passed over to appropriate Client Specific Processing Unit (CSPU) for further processing. The following messages are dropped by MCA:

- Messages that do not have a selected source ID
- IDLE messages
- Messages with number of beats larger than defined beats for data-write messages

MCA has the capability of providing one fully concatenated message to a CSPU each clock.

10.3.3 Client Specific Processing (CSP)

The Client Specific Processing (CSP) block receives the messages from MCA. The received message are processed after extracting all the relevant fields. Different types of message from various clients are received and processed differently based on their type by a specialized unit called the Client Specific Processing Unit (CSPU).

The CSPU receives concatenated data write messages (including synchronization messages) from MCA and provides the following for the Out of Range Check (ORC) and Address Tag Check (ATC) logic to perform filter checks:

- Reconstructed address
- Data value
- Source ID
- Master ID
- Timestamp

CSPU also confirms that each message is a data write message. All other message types are dropped.

There are two CSPU types, one associated with Core Nexus clients and another associated with NXMC clients. Each CSPU simultaneously supports data write messages with up to four different source IDs. Data write with synchronization message is always required to start an address thread for a particular source ID, then each subsequent data write message associated with that source ID contains an encoded address. This table shows the CSPU type encoding to be used while mapping the source IDs to a CSPU type using SRC_MAP_CTLR.

Table 10-1. CSPU encoding

CSTU type	Encoding
Standard	01
NXMC	10

10.3.3.1 Input message formats

DWPU supports various message formats from different source clients. These formats are further processed by the CSPU. The following message formats are supported by the CSPUs, as also described in details the sub-sections:

- Standard data write message format (SDWM)—Includes sync and non-sync type. This is used by most of the clients, i.e., e200 peripheral and e200 main cores.
- AHBMM (modified) data write message format (MDWM)—Includes both sync and non-sync type of messages. Used by NXMC clients only.

10.3.3.1.1 Standard data trace write message format

The standard data trace write messages are of two types, sync and non-sync. The non-sync type contains the data write value and the address of the write access, relative to the previous data trace message. The address and DATA field is considered variable. Data write message information is messaged out in the format shown below. The SRC field values are described in the device-specific chapter that describes how the clients are configured.

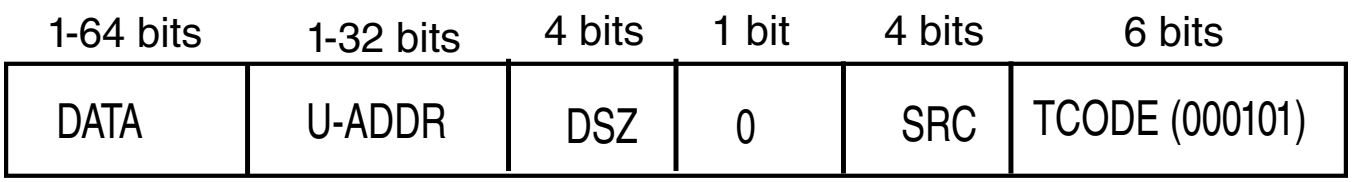


Figure 10-3. Non-sync data trace write message format

The TSTAMP field not shown is an optional variable field upto 26 bits, is inserted optionally by the core clients based on controls received from the BD NAR.

The second type is the DWM synchronization message, whose format is exactly same as above but with full-address (without leading zeros) instead of the relative address. This is to ensure that development tools fully synchronize with data trace regularly.

Synchronization messages provide a reference address for subsequent DWMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write with synchronization messages is shown below.

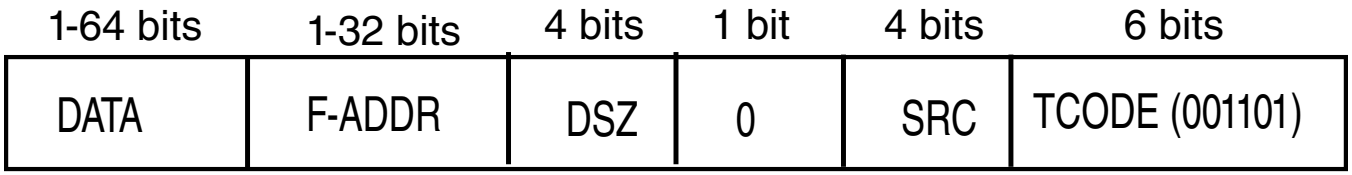


Figure 10-4. Sync data trace write message format

The table below shows the data size (DSZ) encoding for standard data trace messages.

Table 10-2. Standard DSZ encoding

DSZ	Data size
0000	0-byte
0001	1-byte
0010	2-byte
0011	3-byte
0100	4-byte
0101	5-byte
0110	6-byte
0111	7-byte
1000	8-byte
1001-1111	Reserved

10.3.3.1.2 NXMC (modified) data trace write message format

The NXMC (modified) data trace write non-sync messages contain the data write value and the address of the write access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encoding to either 8, 16, 32 or 64. The TSTAMP field is 26 bits. Data write message information is messaged out in the format shown below. The SRC field values are described in the device-specific chapter that describes how the clients are configured.

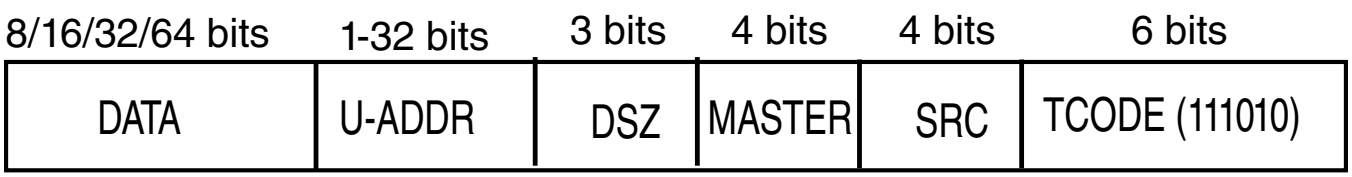


Figure 10-5. Non-sync M-Data trace write message format

The TSTAMP field not shown over here is an optional variable field upto 26 bits, is inserted optionally by the AHBMM clients based on controls received from NAR.

The second type is the MDWM synchronization message, whose format is exactly same as above but with full-address (without leading zeros) instead of the relative address. This is to ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent MDWMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write with synchronization messages is shown below.

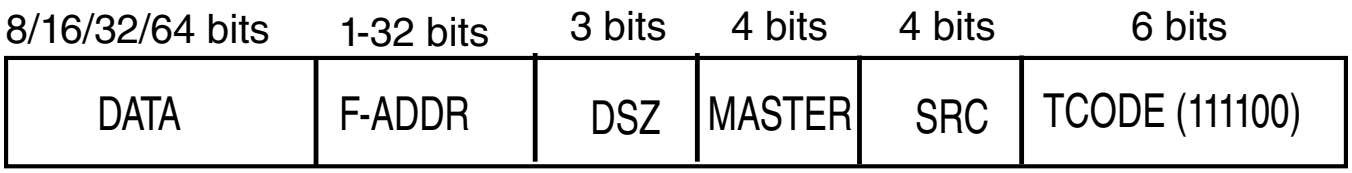


Figure 10-6. Sync M-Data trace write message foramt

The table below shows the data size (DSZ) encoding for NXMC data trace message.

Table 10-3. NXMC DSZ encoding

DSZ	Data Size
001	1-byte
010	2-byte
011	4-byte
100	8-byte
000,101-111	Reserved

10.3.4 Address Tag Check (ATC)

DWPU allows filtering operation on data writes to any address within system RAM and tightly coupled data memory. Each 64-bit word in the supported system RAM or data memory address space corresponds to a single bit in Tag RAM; therefore, the required Tag RAM size is 1/64th of the address space to be filtered. Data write messages within the supported address range are dropped by ATC if the corresponding bit in Tag RAM is not set. The data write messages with the corresponding Tag bit set are passed over to Message Formatter for further processing.

Writes to addresses outside of the supported address range can be configured to be either removed from the trace stream or retained. This option is controlled by a single user accessible control bit. Data reads and other Nexus message formats submitted in the DWPU trace stream are always dropped by DWPU.

Tag RAM is accessible through software (IPS) as well as JTAG interface via PD addressing. All the IPS access is possible in supervisor mode only. The access to Tag RAM is carried out indirectly through three registers: TRC_CTRL, TRC_ADR and TRC_DATA registers. Users should write the TRC_CTRL register followed by the TRC_ADR register before accessing the TRC_DATA register. Any write access to the TRC_ADR register results in a Tag RAM access request. Users should ensure that the Tag RAM request is completed by providing the required TRC_DATA access based on the length of the transfer. In case of block access, the TRC_DATA register is accessed subsequently and user should ensure that the block access length does not cross the region boundary. Deviation in the sequence of register access for TAG RAM access results in errornous and incomplete transfers.

In PD addressing, the configured PD address in the TRC_ADR register is translated to a TAG RAM address using mapping derived from the FAR_SA_CFGn and FAR_RS_CFGn register. The PD address configured in the FAR_SA_CFG0 register is always translated to the zeroth address of Tag RAM.

The equation to convert system/PD address to Tag RAM address where the System/PD address falls within region n is given below:

Tag RAM address = System/PD address[31:8] - Region n base address[31:8] + Total sum of previous region (n-1) size[31:8]

The Tag RAM address for System/PD address that falls on region 2 is given by:

System/PD address[31:8] - Region 2 base address[31:8] + (region 0 size[31:8] + region 1 size[31:8])

This figure shows an example of the Tag RAM organization for 8 Kbyte Tag RAM with PD base address configured.

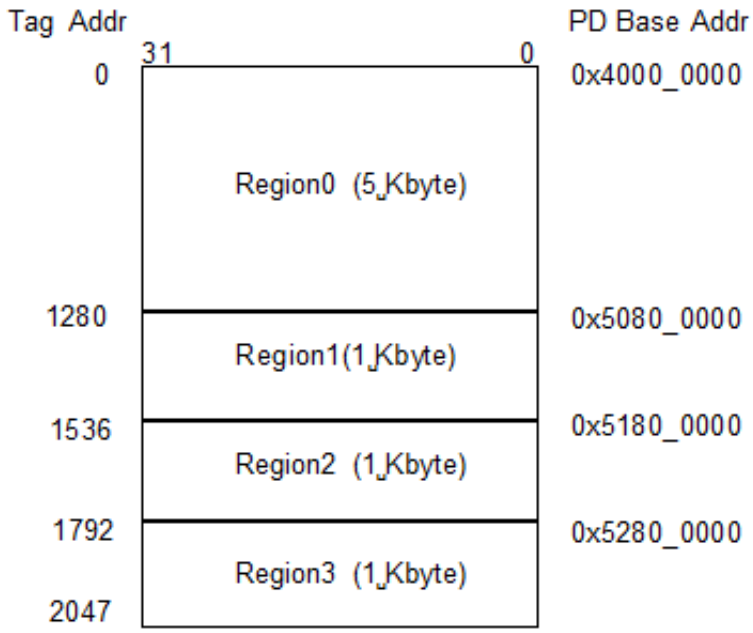


Figure 10-7. Tag RAM organization example

The table below shows an example of address translation from PD addresses to Tag RAM address.

Table 10-4. Address translation example

PD address		Tag RAM address
Start	End	
0x4000_0000	0x4000_00FF	0x0
0x4000_0100	0x4000_01FF	0x1
0x4004_FF00	0x4004_FFFF	0x4FF
0x5080_0000	0x5080_00FF	0x500
0x5180_0000	0x5180_00FF	0x600
0x5280_0000	0x5280_00FF	0x700
0x5280_FF00	0x5280_FFFF	0x7FF

During the data write filtering operation, the system RAM or data memory address is extracted from the input message. It is then compared to ascertain the address space (whether it is within system RAM space or data memory or outside the supported address range). Finally it is translated to Tag RAM address and initiate a read to Tag RAM. From the Tag RAM 32-bit read accessed, the system/PD address bit [7:3] determine the Tag bit position that is associated with the decoded absolute system/PD address. If the corresponding bit is set, then the input message is passed for message formatting else it is dropped by ATC.

This figure shows the data flow for address Tag filtering.

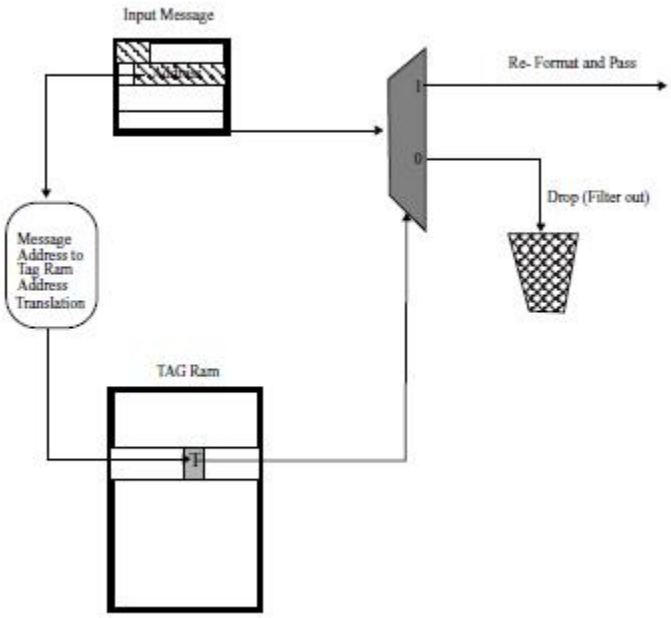


Figure 10-8. Address tag filtering

10.3.5 Out of Range Check (ORC)

In addition to the trace filtering function, the DWPU also supports detection of outside/ inside range writes to RAM locations. Both the data range values and address of RAM locations to be checked are user specified. It also has configurable support to detect target inside or outside data range values specified.

A total of four sets of address and data range comparators are supported, where each set of comparators include an address match comparator, a greater than or equal data value comparator and a less than or equal data value comparator. This figure shows the block diagram for range detect comparators.

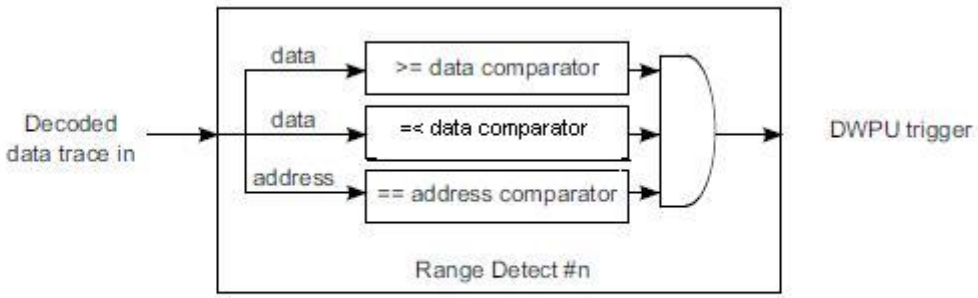


Figure 10-9. Range detect comparators

ORC supports unsigned, signed and floating point data ranges. Signed numbers are represented in 2's complement. 32-bit IEEE-754 single-precision floating-point format is used for floating point representation.

ORC has a configurable option to support for 8/16/32 bit unsigned/signed range detection and 32-bit floating point range comparison. Address comparison is carried out with full the device address. An output signal from DWPU is triggered in response to a detected out of range condition along with a status bit being set in a ORC status register indicating which range comparator has caused the trigger.

10.3.5.1 Comparator capabilities

The primary usage for out of range detect function is to detect out of range conditions on unsigned integer and pointer variables, although it also supports signed and floating point data types comparison. It also has a configurable option to support for 8/16/32 bit unsigned range detection. The unsigned data is directly compared with the stored data ranges.

To support use with signed variables, each data comparator supports 8/16/32 bit signed operation. All the signed numbers are represented in two’s complement. Data comparison happens only when the sign bit matches.

To detect range of floating point variables, each data comparator supports only 32-bit comparison mode. All the floating point numbers are represented in 32-bit IEEE-754 single precision with biased exponent. The floating point data is directly compared with the stored floating point ranges which are also represented in 32-bit IEEE-754 single precision with biased exponent.

10.3.6 Message Formatter (MF)

Message Formatter (MF) module receives message information from ATC and reformats the data write messages to comply with output Nexus message format. A user defined message format is used for sending out the filtered messages as shown below.

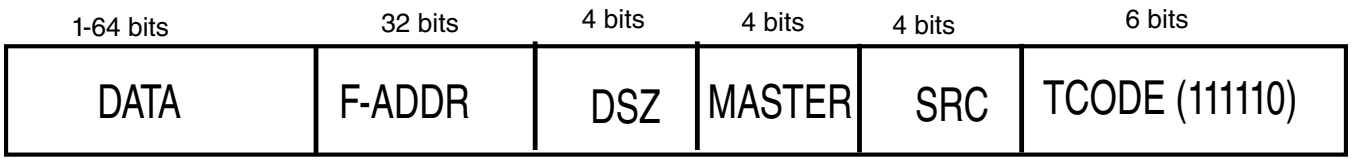


Figure 10-10. DWPU message format

Table 10-5. Message field description

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
DWPU message	6	6	TCODE	fixed	TCODE number = 62
	4	4	SRC	fixed	Source client identifier
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access. For core clients 1111 is inserted. For the NXMC clients, same value is inserted as received.
	4	4	DSZ	fixed	Data size
	32	32	F-ADDR	fixed	Full data write address
	1	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	26	26	TSTAMP	fixed	Timestamp (Optional and omitted if there is no valid timestamp)

10.3.7 Nexus Client Interface (NCI)

The Nexus Client Interface (NCI) sends out the formatted filtered data write messages to BD NAR. It also supports the buffer reallocation to increase the Q size.

10.4 Non-memory mapped registers

This section describes the DWPU Nexus register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. In addition, some register are accessible through the IPS interface which are described in the next section.

The table below shows the JTAG mapped DWPU registers.

Table 10-6. JTAG DWPU registers

Nexus register index	Register	Access	Read address	Write address	Reset value
0x02	Global Control (GLB_CTRL)	R/W	0x04	0x05	0x0000_0002
0x03	Source Mapping Control 0 (SRC_MAP_CTLR0)	R/W	0x06	0x07	0x0000_0000
0x04	Source Mapping Control 1 (SRC_MAP_CTLR1)	R/W	0x08	0x09	0x0000_0000
0x05	Source Mapping Control 2 (SRC_MAP_CTLR2)	R/W	0x0A	0x0B	0x0000_0000
0x0A	Filter Address Range Start Address 0 (FAR_SA_CFG0)	R/W	0x14	0x15	0x0000_0000
0x0B	Filter Address Range Region Size 0 (FAR_RS_CFG0)	R/W	Ox16	0x17	0x0000_0000
0x0C	Filter Address Range Start Address 1 (FAR_SA_CFG1)	R/W	0x18	0x19	0x0000_0000
0x0D	Filter Address Range Region Size 1 (FAR_RS_CFG1)	R/W	0x1A	0x1B	0x0000_0000
0x0E	Filter Address Range Start Address 2 (FAR_SA_CFG2)	R/W	0x1C	0x1D	0x0000_0000
0x0F	Filter Address Range Region Size 2 (FAR_RS_CFG2)	R/W	0x1E	0x1F	0x0000_0000
0x10	Filter Address Range Start Address 3 (FAR_SA_CFG3)	R/W	0x20	0x21	0x0000_0000
0x11	Filter Address Range Region Size 3 (FAR_RS_CFG3)	R/W	0x22	0x23	0x0000_0000
0x1A	Tag RAM Control (TRC_CTRL)	R/W	0x34	0x35	0x0000_0000
0x1B	Tag RAM Configuration Address (TRC_ADR)	R/W	0x36	0x37	0x0000_0000
0x1C	Tag RAM Configuration Data (TRC_DATA)	R/W	0x38	0x39	0x0000_0000
0x20	Out of Range Check Control 1 (ORC_CTRL1)	R/W	0x40	0x41	0x0000_0000
0x21	Out of Range Check Control 2 (ORC_CTRL2)	R/W	0x42	0x43	0x0000_0000
0x22	Global Status (GLB_STAT)	R	0x44	0x45	0x0004_0000
0x24	Out of Range Configuration Address 0 (ORC_CFG0_ADR)	R/W	0x48	0x49	0x0000_0000
0x25	Out of Range Configuration Data High 0 (ORC_CFG0_DH)	R/W	0x4A	0x4B	0x0000_0000
0x26	Out of Range Configuration Data Low 0 (ORC_CFG0_DL)	R/W	0x4C	0x4D	0x0000_0000
0x33	Out of Range Configuration Data Mask 0 (ORC_CFG0_DM)	R/W	0x4E	0x4F	0x0000_0000
0x28	Out of Range Configuration Address 1 (ORC_CFG1_ADR)	R/W	0x50	0x51	0x0000_0000
0x29	Out of Range Configuration Data High 1 (ORC_CFG1_DH)	R/W	0x52	0x53	0x0000_0000
0x2A	Out of Range Configuration Data Low 1 (ORC_CFG1_DL)	R/W	0x54	0x55	0x0000_0000
0x33	Out of Range Configuration Data Mask 1 (ORC_CFG1_DM)	R/W	0x56	0x57	0x0000_0000
0x2C	Out of Range Configuration Address 2 (ORC_CFG2_ADR)	R/W	0x58	0x59	0x0000_0000
0x2D	Out of Range Configuration Data High 2 (ORC_CFG2_DH)	R/W	0x5A	0x5B	0x0000_0000
0x2E	Out of Range Configuration Data Low 2 (ORC_CFG2_DL)	R/W	0x5C	0x5D	0x0000_0000

Table continues on the next page...

Table 10-6. JTAG DWPU registers (continued)

Nexus register index	Register	Access	Read address	Write address	Reset value
0x33	Out of Range Configuration Data Mask 2 (ORC_CFG2_DM)	R/W	0x5E	0x5F	0x0000_0000
0x30	Out of Range Configuration Address 3 (ORC_CFG3_ADR)	R/W	0x60	0x61	0x0000_0000
0x31	Out of Range Configuration Data High 3 (ORC_CFG3_DH)	R/W	0x62	0x63	0x0000_0000
0x32	Out of Range Configuration Data Low 3 (ORC_CFG3_DL)	R/W	0x64	0x65	0x0000_0000
0x33	Out of Range Configuration Data Mask 3 (ORC_CFG3_DM)	R/W	0x66	0x67	0x0000_0000

10.4.1 Register descriptions

These sections describe the JTAG accessible registers.

10.4.1.1 Global Control (GLB_CTRL)

Index: 0x02

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FARE3	FARE2	FARE1	FARE0	TSC	0	0	0	0	0	ORC	SFT_RST	MEN
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 10-7. GLB_CTRL field descriptions

Field	Description
MEN	Module Enable. 0 DWPU disabled 1 DWPU enabled
SFT_RST	Active low soft reset. Once the soft reset is enabled, user is expected to remove the reset by disabling the soft reset 0 Enabled soft reset 1 Disable soft reset
ORC	0 Data writes outside the identified RAM areas are flushed

Table continues on the next page...

Table 10-7. GLB_CTRL field descriptions (continued)

Field	Description	
	1	Data writes outside the identified RAM areas are passed
TSC	0	Pass incoming timestamp and include in output message format (requires more trace bandwidth or overlay RAM locations)
	1	Drop timestamp from incoming messages
FARE[0–3]	0	Filter Address Range n Configuration (FAR_SA_CFGn, FAR_RS_CFGn) disabled.
	1	Filter Address Range n Configuration (FAR_SA_CFGn, FAR_RS_CFGn) enabled.

10.4.1.2 Source Mapping Control n (SRC_MAP_CTLRn)

Index:
n=0 0x03
n=1 0x04
n=2 0x05

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN1	0	0	0	0	CSP_TYPE1			SRC_ID1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EN0	0	0	0	0	CSP_TYPE0			SRC_ID0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-8. SRC_MAP_CTLRn field descriptions

Field	Description	
SRC_ID0	Source ID selected	
CSP_TYPE0	CSPU Type to which the SRC_ID0 is mapped	
	000	Unmapped (to be dropped by CSP)
	001	Standard message
	010	NXMC message
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
EN0	0	SRC_ID0 field is not used or mapped to a CSPU. Messages with unmapped source IDs are flushed
	1	SRC_ID and CSP_TYPE0 are used for selecting and mapping data write messages

Table continues on the next page...

Table 10-8. SRC_MAP_CTLRn field descriptions (continued)

Field	Description
SRC_ID1	Source ID selected
CSP_TYPE1	CSPU Type to which the SRC_ID1 is mapped 000 Unmapped (to be dropped by CSP) 001 Standard message 010 NXMC message 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
EN1	0 SRC_ID0 field is not used or mapped to a CSPU. Messages with unmapped source IDs are flushed 1 SRC_ID and CSP_TYPE0 are used for selecting and mapping data write messages

10.4.1.3 Filter Address Range Start Address n (FAR_SA_CFGn)

Index:
n=0 0x0A
n=1 0x0C
n=2 0x0E
n=3 0x10

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SA																								0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-9. FAR_SA_CFGn field descriptions

Field	Description
SA	Start PD address for the region, should be at 265-byte boundaries

10.4.1.4 Filter Address Range Region Size n (FAR_RS_CFGn)

Index:
n=0 0x0B
n=1 0x0D
n=2 0x0F
n=3 0x11

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	RS												0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-10. FAR_RS_CFGn field descriptions

Field	Description
RS	Region size in 256-bytes

10.4.1.5 Tag RAM Control (TRC_CTRL)

Index: 0x1A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BAL												0	0	0	0	ERR	EXC	PROG	INV	0	DSM			0	BAE	CMD	
W																					w 1c	w 1c		w 1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-11. TRC_CTRL field descriptions

Field	Description
CMD	00 No read or write access to Tag RAM 01 Read access to Tag RAM. Any write to TRC_ADR register results in read access to the Tag RAM. Read data is stored in the TRC_DATA register. 10 Write access to Tag RAM. Any write to TRC_ADR register results in write access request to Tag RAM and subsequent TRC_DATA register write results in actual data to be written into Tag RAM addressed by TRC_ADR register 11 Reserved
BAE	Block Access Enable. Block access is carried out with 32-bit Data Size Mode only. 0 Block Access disable 1 Block Access enable

Table continues on the next page...

Table 10-11. TRC_CTRL field descriptions (continued)

Field	Description
DSM	Data Size Mode. Valid only for writes, all reads are fixed 32 bits only. 000 Word (32 bit) Access to the Tag RAM. 001 Half-Word (16-bit) access to Tag RAM. 16-bit LSBs of the TRC_DATA is used. 010 Byte (8-bit) access to the Tag RAM. 8-bit LSBs of the TRC_DATA is used. 100 Bit Access to the Tag RAM. LS bit of the TRC_DATA is used.
INV	Invalid JTAG access. Generated when the targeted system/PD address in TRC_ADR register falls outside the configured Tag Rsm region. 0 No error 1 Invalid address
PROG	JTAG access to Tag RAM in progress. 0 No Tag RAM access by JTAG 1 Tag RAM access by JTAG is in progress
EXC	Excess Tag RAM access by JTAG. Generated when JTAG access to Tag RAM exceeds the configured access length. 0 No excess access to Tag RAM 1 Exceeds the configured access length
ERR	Error JTAG access. Generated when read/write access to Tag RAM before getting the grant. Also generated on new JTAG Tag RAM access request before completing the previous access. **NOTE:** This field is not present in the IPS mapped version of this register. 0 No error 1 Error
BAL	Block Access Length. Should not cross the region boundary. Valid only when BAE = 1

10.4.1.6 Tag RAM Configuration Address (TRC_ADR)

Index: 0x1B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																													0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-12. TRC_ADR field descriptions

Field	Description
ADDR	PD Address. It should be aligned depending on Data Size mode field. DSM = 000, 256 byte aligned DSM = 001, 128 byte aligned DSM = 010, 64 byte aligned DSM = 100, 8 byte aligned (default)

Table 10-12. TRC_ADR field descriptions

Field	Description
	Note: If the PD address is not aligned to the DSM, this results in a erroneous data read/write.

10.4.1.7 Tag RAM Configuration Data (TRC_DATA)

Index: 0x1C

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-13. TRC_DATA field descriptions

Field	Description
DATA	Reflets the content of Tag RAM on Tag RAM read access. Also provides data for Tag RAM write access

10.4.1.8 Out of Range Check Control 1 (ORC_CTRL1)

Index: 0x20

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	CMP_MD3		CMP_MD2		CMP_MD1		CMP_MD0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	TOR3	TOR2	TOR1	TOR0	0	0	0	0	RDCE3	RDCE2	RDCE1	RDCE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-14. ORC_CTRL1 field descriptions

Field	Description	
RDCE0	0	Disable Range Detect comparator 0
	1	Enable Range Detect comparator 0
RDCE1	0	Disable Range Detect comparator 1
	1	Enable Range Detect comparator 1
RDCE2	0	Disable Range Detect comparator 2

Table continues on the next page...

Table 10-14. ORC_CTRL1 field descriptions (continued)

Field	Description
	1 Enable Range Detect comparator 2
RDCE3	0 Disable Range Detect comparator 3 1 Enable Range Detect comparator 3
TOR0	Comparison Mode for Comparator 0 0 Trigger is generated if data write value is within the range ORC_CFG0_DL ≤ data_wr_value ≤ ORC_CFG0_DH 1 Trigger is generated if data write value is outside the range data_wr_value < ORC_CFG0_DL or data_wr_value > ORC_CFG0_DH
TOR1	Comparison Mode for Comparator 1 0 Trigger is generated if data write value is within the range ORC_CFG1_DL ≤ data_wr_value ≤ ORC_CFG1_DH 1 Trigger is generated if data write value is outside the range data_wr_value < ORC_CFG1_DL or data_wr_value > ORC_CFG1_DH
TOR2	Comparison Mode for Comparator 2 0 Trigger is generated if data write value is within the range ORC_CFG2_DL ≤ data_wr_value ≤ ORC_CFG2_DH 1 Trigger is generated if data write value is outside the range data_wr_value < ORC_CFG2_DL or data_wr_value > ORC_CFG2_DH
TOR3	Comparison Mode for Comparator 3 0 Trigger is generated if data write value is within the range ORC_CFG3_DL ≤ data_wr_value ≤ ORC_CFG3_DH 1 Trigger is generated if data write value is outside the range data_wr_value < ORC_CFG3_DL or data_wr_value > ORC_CFG3_DH
CMP_MD0	00 8 bit data comparison for comparator 0. 01 16 bit data comparison for comparator 0. 10 32 bit data comparison for comparator 0. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.
CMP_MD1	00 8 bit data comparison mode for comparator 1. 01 16 bit data comparison mode for comparator 1. 10 32 bit data comparison mode for comparator 1. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.
CMP_MD2	00 8 bit data comparison mode for comparator 2. 01 16 bit data comparison mode for comparator 2. 10 32 bit data comparison mode for comparator 2. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.
CMP_MD3	00 8 bit data comparison mode for comparator 3. 01 16 bit data comparison mode for comparator 3.

Table continues on the next page...

Table 10-14. ORC_CTRL1 field descriptions (continued)

Field	Description
	10 32 bit data comparison mode for comparator 3. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.

10.4.1.9 Out of Range Check Control 2 (ORC_CTRL2)

Index: 0x21

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DTYP3		DTYP2		DTYP1		DTYP0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	G	0	0	0	0	0	0	0	0
W								T M								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-15. ORC_CTRL2 field descriptions

Field	Description
GTM	Generate Trigger on Mismatch 0 Disable trigger generation on data-size mismatch 1 Enable trigger generation on data-size mismatch Mismatch happens if the data write size is less than comparator size (CMP_MD)
DTYP0	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
DTYP1	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
DTYP2	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
DTYP3	00 Unsigned data type comparison

Table continues on the next page...

Table 10-15. ORC_CTRL2 field descriptions (continued)

Field	Description
	01 Signed data type comparison 10 Floating data type comparison 11 Reserved

10.4.1.10 Global Status (GLB_STAT)

Index: 0x22

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIDLE	0	TRSE
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	DSM3	DSM2	DSM1	DSM0	0	0	0	0	TRDC3	TRDC2	TRDC1	TRDC0
W					w1c	w1c	w1c	w1c					w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-16. GLB_STAT field descriptions

Field	Description
DIDLE	DWPU module idle indication 1 Indicates that there is no more processing going on inside the module, or the all the messages have been completely processed and sent out.
TRSE	Tag RAM Size Error 1 Configured PD region for ATC exceeds the maximum Tag RAM size.
TRDC0	1 Data range comparison for ORC_CFG0_ADR was successful
TRDC1	1 Data range comparison for ORC_CFG1_ADR was successful
TRDC2	1 Data range comparison for ORC_CFG2_ADR was successful
TRDC3	1 Data range comparison for ORC_CFG3_ADR was successful
DSM0	1 Received data write trace size is smaller than comparator size (CMP_MP0)
DSM1	1 Received data write trace size is smaller than comparator size (CMP_MP1)
DSM2	1 Received data write trace size is smaller than comparator size (CMP_MP2)
DSM3	1 Received data write trace size is smaller than comparator size (CMP_MP3)

10.4.1.11 Out of Range Configuration Address n (ORC_CFGn_ADR)

Index:
n=0 0x24
n=1 0x28
n=2 0x2c
n=3 0x30

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-17. ORC_CFGn_ADR field descriptions

Field	Description
ADDR	Address value for the Data range comparison.

10.4.1.12 Out of Range Configuration Data High n (ORC_CFGn_DH)

Index:
n=0 0x25
n=1 0x29
n=2 0x2D
n=3 0x31

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_H																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-18. ORC_CFGn_DH field descriptions

Field	Description
DATA_H	Data High value for the comparator.

10.4.1.13 Out of Range Configuration Data Low n (ORC_CFGn_DL)

Index:
n=0 0x26
n=1 0x2A
n=2 0x2E
n=3 0x32

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_L																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-19. ORC_CFGn_DL field descriptions

Field	Description
DATA_L	Data low value for the comparator.

10.4.1.14 Out of Range Configuration Data Mask n (ORC_CFGn_DM)

Index:
n=0 0x27
n=1 0x2B
n=2 0x2F
n=3 0x33

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-20. ORC_CFGn_DM field descriptions

Field	Description
D_MASK	Data Mask value for the comparator (bit wise). 0 Data comparison not masked 1 Data comparison masked

10.5 Memory mapped registers

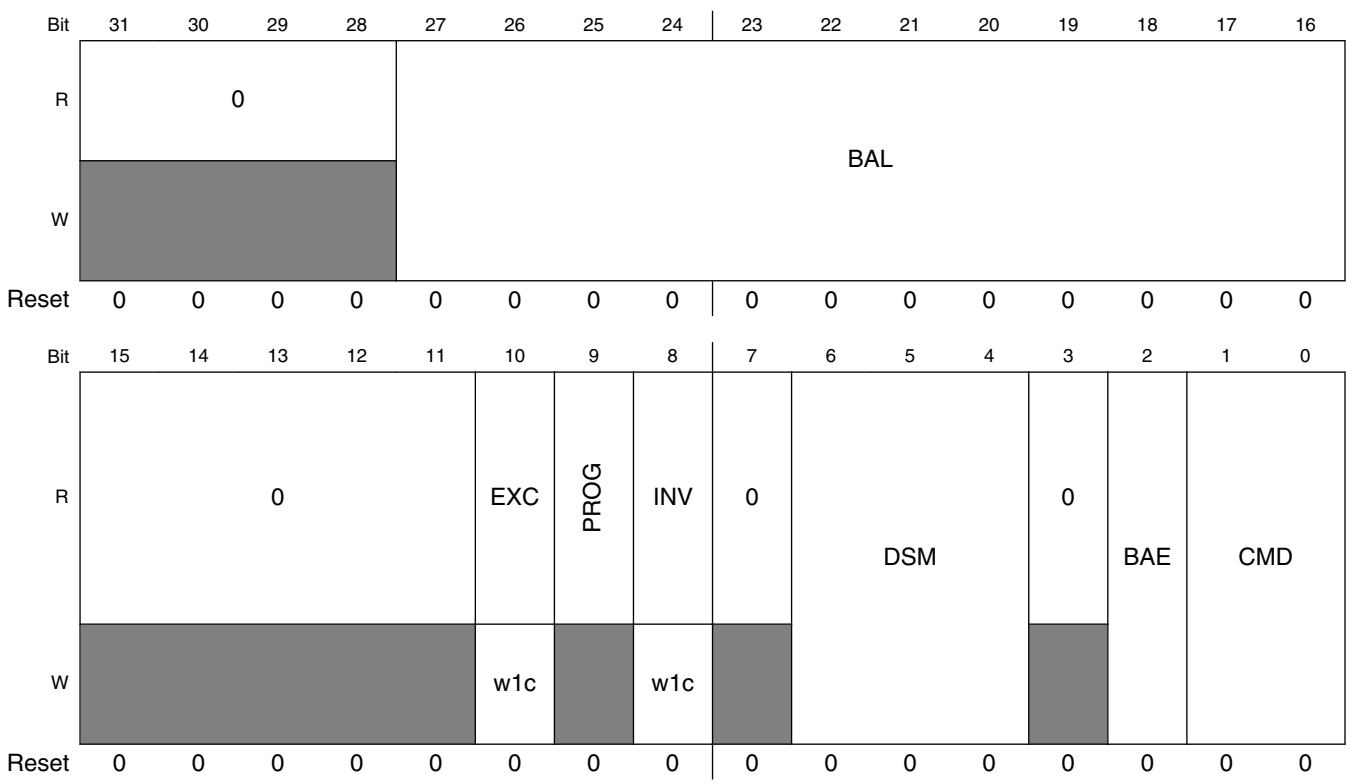
DWPU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Tag RAM Control (DWPU_TRC_CTRL)	32	R/W	0000_0000h	10.5.1/181
4	Tag RAM Configuration Address (DWPU_TRC_ADR)	32	R/W	0000_0000h	10.5.2/183
8	Tag RAM Configuration Data (DWPU_TRC_DATA)	32	R/W	0000_0000h	10.5.3/183

10.5.1 Tag RAM Control (DWPU_TRC_CTRL)

Tag RAM Control register provides the configuration for Tag RAM access. It supports byte write IPS access. It also has a status bit to indicate the status of the Tag RAM access.

Address: 0h base + 0h offset = 0h



DWPU_TRC_CTRL field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

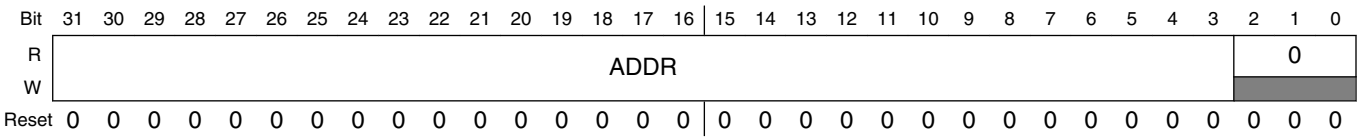
DWPU_TRC_CTRL field descriptions (continued)

Field	Description
27–16 BAL	Block Access Length. Should not cross region boundary. Valid only when BAE = 1
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EXC	Excess Tag RAM access by IPS Generates when the IPS acsess to Tag RAM exceeds the configured length 0 No excess access to Tag RAM 1 Exceeds the configured access length
9 PROG	IPS access to Tag RAM in progress 0 No Tag RAM access by IPS 1 Tag RAM access by IPS is in progress
8 INV	Invalid IPS access Generated when the targetted PD/system address in TRC_ADR register falls outside the configured Tag RAM region 0 No error 1 Invalid address
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 DSM	Data Size Mode Valid only for writes, all reads are fixed 32 bits only. 000 Word (32 bit) access to the Tag RAM. 001 Half-Word (16 bit) access to the Tag RAM. 16 bit LSBs of the TRC_DATA is used. 010 Byte (8 bit) access to the Tag RAM. 8 bit LSBs of the TRC_DATA is used. 100 Bit access to the Tag RAM. LS bit of the TRC_DATA is used.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 BAE	Block Access Enable. Block access is carried out with 32-bit Data Size Mode only. 0 Block Access disable 1 Block Access enable
CMD	Read/Write access 00 No read or write access to Tag RAM 01 Read access to Tag RAM. Any write to TRC_ADR register results in read access to the Tag RAM. Read data is stored in the TRC_DATA register. 10 Write access to Tag RAM. Any write to TRC_ADR register results in write access request to Tag RAM and subsequent TRC_DATA register write results in actual data to be written into Tag RAM addressed by TRC_ADR register. 11 Reserved

10.5.2 Tag RAM Configuration Address (DWPU_TRC_ADR)

Tag RAM Configuration Address provides the system address for Tag RAM access. Any write to this register initiates a Tag RAM access request at the address which is being translated from system address configured. It is always accessed in 32-bit only.

Address: 0h base + 4h offset = 4h



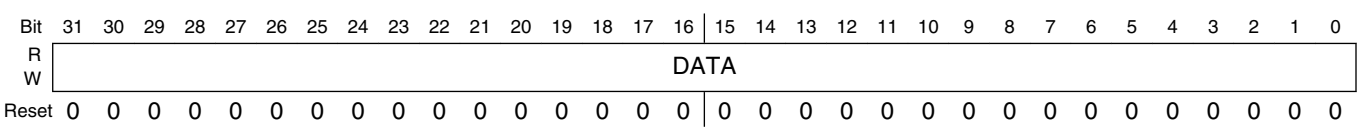
DWPU_TRC_ADR field descriptions

Field	Description
31–3 ADDR	PD Address. It should be aligned depending on Data Size mode field. DSM = 000, 256 byte aligned DSM = 001, 128 byte aligned DSM = 010, 64 byte aligned DSM = 100, 8 byte aligned (default) If the PD address is not aligned to the DSM, this results in a erroneous data read/write.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.5.3 Tag RAM Configuration Data (DWPU_TRC_DATA)

Tag RAM Configuration read/write data register. Generate ips_xfer_err on accessing this register before configuring TRC_ADDR register. Also on read access to this register while Tag RAM write is in progress and vice vera. Access is always 32-bit. During write, TRC_ADR.DSM determines which part of TRC_DATA should be written irrespective of byte enable

Address: 0h base + 8h offset = 8h



DWPU_TRC_DATA field descriptions

Field	Description
DATA	Reflets the content of Tag RAM on Tag RAM read access. Also provides data for Tag RAM write access

Chapter 11
Buddy Device—Nexus Read Write Access (RWA)

11.1 Introduction

The Nexus Read Write Access (RWA) module allows an external tool connected to the Emulation and Debug Device (ED) to read or write the Buddy Die calibration and trace memory via the JTAG interface. All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). The RWA module supports single as well as block reads and writes.

The RWA module is enabled by loading a single instruction (ACCESS_AUX_RWA_BD = 0x23) into the JTAG Instruction Register (IR). After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE) into the JTAG IR, as described in Nexus register access via JTAG. The RWA module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. The RWA module is also disabled if a Power-on-Reset (POR) event occurs. When disabled, the RWA module registers are not available for (32-bit or less) reads or writes.

11.2 Register definition

This table details the register map for the RWA module.

Table 11-1. RWA register map

Register	Nexus access opcode	Read/ write	Read address	Write address
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15

11.2.1 Read/write access control/status (RWCS)

The Read/write access control/status register provides control and status for the RWA module as shown in this table.

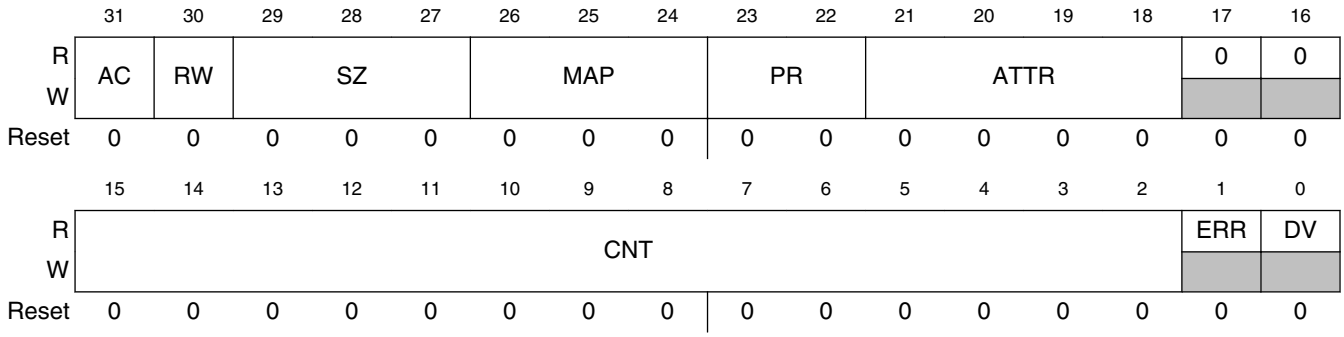


Table 11-2. RWCS fields

Field	Description	
31 AC	Access Control.	
	0	End access/ Access has completed
	1	Start access
30 RW	Read/Write Select.	
	0	Read access
	1	Write access
29:27 SZ	Word Size.	
	000	8-bit (byte)
	001	16-bit (half-word)
	010	32-bit (word)
	011	Reserved (default to word)
	100	Reserved (default to word)
	101	Reserved (default to word)
	110	Reserved (default to word)
	111	Reserved (default to word)
26:24 MAP	MAP Select.	
	000	Primary memory map
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 11-2. RWCS fields (continued)

Field	Description
23:22 PR	Read/Write Access Priority. 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
21:18 ATTR	Access Attributes. RWA accesses are run as privileged data non-cacheable accesses by default, and drive the p_d_hprot[5:0] bus access attributes to 6'b000011 accordingly. This field is provided to allow a portion of these default values to be modified when performing read or write accesses using the RWA as shown in Table 11-4. hprot[1] must be set to one. 0xxx Reserved 1xxx Reserved x0xx p_d_hprot[4] driven to 0 for accesses x1xx p_d_hprot[4] driven to 1 for accesses xx0x p_d_hprot[3] driven to 0 for accesses xx1x p_d_hprot[3] driven to 1 for accesses xxx0 p_d_hprot[2] driven to 0 for accesses xxx1 p_d_hprot[2] driven to 1 for accesses
17:16	Reserved for future functionality.
15:2 CNT	Access Control Count. Number of accesses of word size SZ.
1 ERR	Read/Write Access Error (see Table 11-3).
0 DV	Read/Write Access Data Valid (see Table 11-3).

Table 11-3. Read/Write access status bit encoding

Read action	Write action	ERR	DV
Read access has not completed	Write access completed without error	0	0
Read access error has occurred	Write access error has occurred	1	0
Read access completed without error	Write access has not completed	0	1
Not allowed	Not allowed	1	1

Table 11-4. Protection encoding

p_hprot[4]	p_hprot[3]	p_hprot[2]	Transfer type
0	0	0	Cache-inhibited
0	0	1	Guarded, not cache-inhibited
0	1	0	Reserved
0	1	1	Reserved

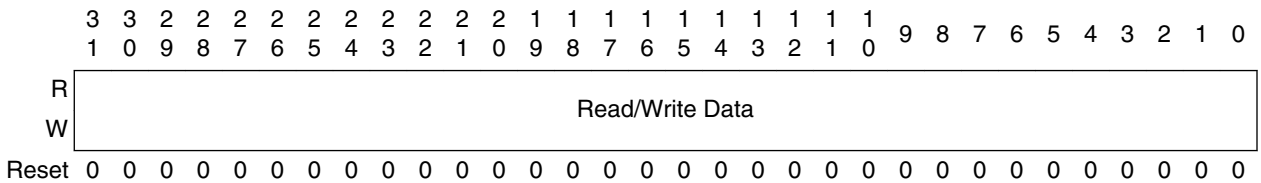
Table continues on the next page...

Table 11-4. Protection encoding (continued)

p_hprot[4]	p_hprot[3]	p_hprot[2]	Transfer type
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Cachable, writethrough
1	1	1	Cachable, writeback

11.2.2 Read/write access data (RWD)

The Read/write access data register (RWD) provides the data to/from memory when initiating a read or a write access.



Read/write accesses to the memory require that the debug firmware properly retrieve/ place the data in the RWD. This table shows the proper placement of data into the RWD.

Table 11-5. RWD data placement for transfers

Transfer size and byte offset	RWA(2:0)	RWCS[SZ]	RWD[1]			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X

1. X indicates byte lanes with valid data; — indicated bytes lanes with unused data.

This table shows the mapping of RWD bytes to byte lanes of the read and write data buses.

Table 11-6. RWD byte lane mapping

Transfer size and byte offset	RWA(2:0)	RWD[1]			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	XBAR[7:0]
Byte @001	0 0 1	—	—	—	XBAR[15:8]
Byte @010	0 1 0	—	—	—	XBAR[23:16]

Table continues on the next page...

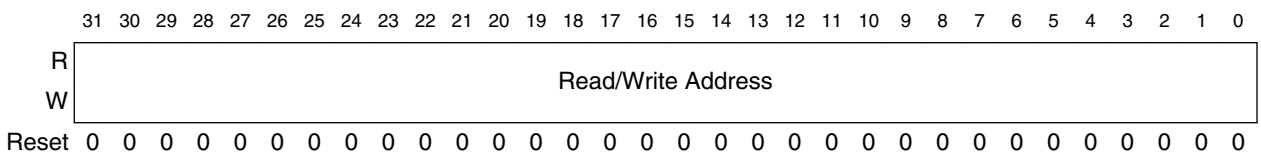
Table 11-6. RWD byte lane mapping (continued)

Transfer size and byte offset	RWA(2:0)	RWD[1]			
		31:24	23:16	15:8	7:0
Byte @011	0 1 1	—	—	—	XBAR[31:24]
Byte @100	1 0 0	—	—	—	XBAR[39:32]
Byte @101	1 0 1	—	—	—	XBAR[47:40]
Byte @110	1 1 0	—	—	—	XBAR[55:48]
Byte @111	1 1 1	—	—	-	XBAR[63:56]
Half @000	0 0 0	—	—	XBAR[15:8]	XBAR[7:0]
Half @010	0 1 0	—-	—	XBAR[31:24]	XBAR[23:16]
Half @100	1 0 0	—-	—	XBAR[47:40]	XBAR[39:32]
Half @110	1 1 0	—	—	XBAR[63:56]	XBAR[55:48]
Word @000	0 0 0	XBAR[31:24]	XBAR[23:16]	XBAR[15:8]	XBAR[7:0]
Word @100	1 0 0	XBAR[63:56]	XBAR[55:48]	XBAR[47:40]	XBAR[39:32]

1. — indicates byte lanes that contain unused data

11.2.3 Read/write access address (RWA)

The Read/write access address register provides the address to be accessed when initiating a read or a write access.



11.3 Nexus register access via JTAG

After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE = 0x7C) into the JTAG Instruction Register (IR). Once the CORE_ENABLE instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 11-1.

Reading or writing of a RWA module register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the RWA module register to be accessed by providing an index and the direction (read/write) as shown in Table 11-1. This is achieved by loading an 8-bit value into the JTAG data register (DR).

This register has the following format:

(7bits)	(1 bit)
Nexus access opcode	R/W

Table 11-7. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 11-1
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

11.4 Memory access examples

The following subsections describe the steps which are required to access memory.

11.4.1 Single write access

1. Initialize the RWA register through the JTAG. Configure as follows:
   - Write Address = 32h'xxxxxxxx (write address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
   - Access Control (AC) = 1b'1 (to indicate start access)
   - Map Select (MAP) = 3b'000 (primary memory map)
   - Access Priority (PR) = 2b'00 (lowest priority)
   - Read/Write (RW) = 1b'1 (write access)

- Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
- Access Count (CNT) = 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. Initialize the RWD register through the JTAG. Configure as follows:
   - Write Data = 32h'xxxxxxxx (write data)
4. The RWA module then transfers the data value from the data buffer RWD register to the memory mapped address in the RWA register. When the access has completed without error (ERR = 1'b0), the RWA clears the DV bit in the RWCS register. This indicates that the device is ready for the next access.

11.4.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the first data value from the RWD register to the memory mapped address in the RWA register. When the transfer has completed without error (ERR=1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates it is ready for the next access.
3. Repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the DV bit within the RWCS is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

11.4.3 Single read access

1. Initialize the RWA register through the JTAG. Configure as follows:
   - Read Address = 32h'xxxxxxxx (read address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
   - Access Control (AC) = 1b'1 (to indicate start access)
   - Map Select (MAP) = 3b'000 (primary memory map)
   - Access Priority (PR) = 2b'00 (lowest priority)
   - Read/Write (RW) = 1b'0 (read access)
   - Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) = 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. The RWA module then transfers the data to the RWD register. When the transfer is completed without error (ERR = 1'b0), the RWA module sets the DV bit in the RWCS register. This indicates that the RWA module is ready for the next access.
4. The data can then be read from the RWD register through the JTAG.

11.4.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the data to the RWD register. When the transfer has completed without error (ERR = 1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates that the device is ready for the next access.
3. The data can then be read from the RWD register through JTAG.

4. Repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the DV bit within the RWCS register is set to indicate the end of the block read access.

Note

The data values must be shifted out 32-bits at a time LSB first. The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

Chapter 12
Buddy Device – LVDS Fast Asynchronous Serial Transmission (LFAST) – High Speed debug

12.1 Introduction

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in slave only mode, enabling high speed debug communications through the JTAG port.

12.2 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

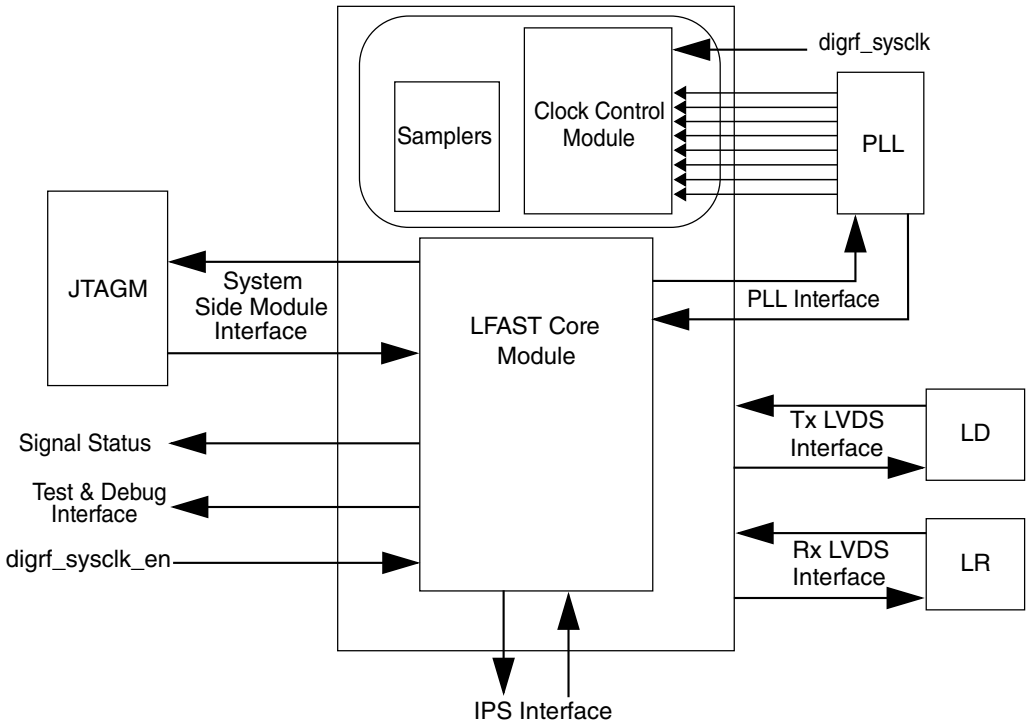


Figure 12-1. LFAST block diagram

12.3 External signals

LFAST is a six pin interface with the following signals after enabling high speed debug mode:

- lfast_sysclk_en
    - Interface enable signal in LFAST Slave Only mode
- lfast_sysclk
    - Reference clock of the LFAST master and slave
- txdatap/txdatan
    - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
    - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface with maximum data rate of 320 Mbps.

12.3.1 LFAST operating data rates

Table 12-1. Operating modes for LFAST interface

Interface	Tx/Rx Data Rate
LFAST Transmit	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps
LFAST Receive	6.5 Mbps / 5 Mbps or 312 Mbps or 320 Mbps

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low and data rates 312 Mbps and 320 Mbps is referred to as high data rates.

12.4 LFAST frame structure

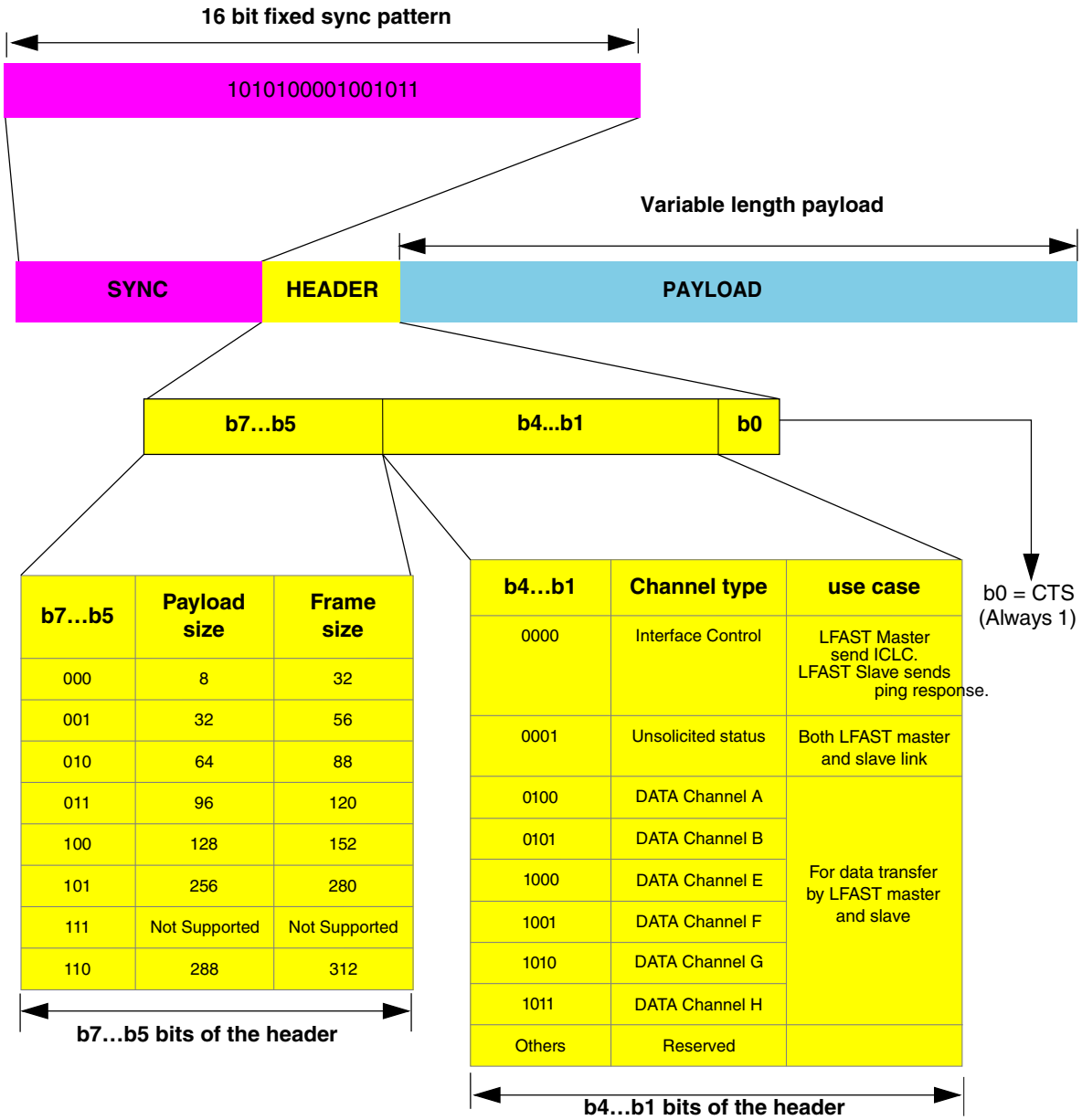
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See Figure 12-2 for details of the LFAST frame.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	Not Supported	Not Supported
110	288	312

b4…b1	Channel type	use case
0000	Interface Control	LFAST Master send ICLC. LFAST Slave sends ping response.
0001	Unsolicited status	Both LFAST master and slave link
0100	DATA Channel A	For data transfer by LFAST master and slave
0101	DATA Channel B	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 12-2. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

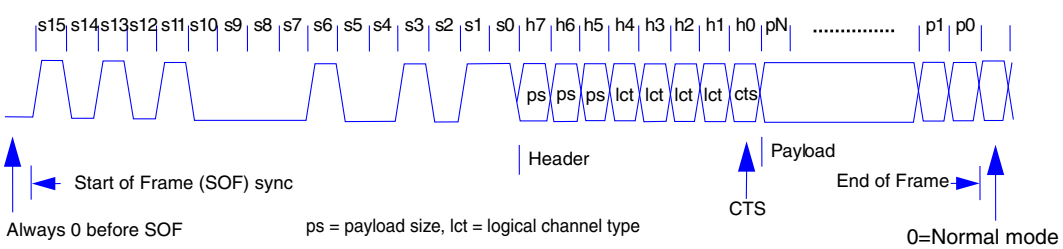


Figure 12-3. Serial frame structure

The frame structure is shown in Figure 12-3 and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in Table 12-2 :

Table 12-2. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	288	312
111	—	—

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type.
- **Header - (b0):** CTS on the both LFAST master and slave devices. In slave only mode the CTS field will always be 1.
- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

12.5 Features

- Supports slave only mode.
- Supports asynchronous data transfer at data rates up to a maximum of 320 Mbps.

- Transmits and receives data, ICLC and unsolicited frames.
- Receives ICLC frames
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation .
- Supports for detection of unsupported channel number and unsupported payload size.

12.6 Memory map and register definition

All registers are 32 bits wide.

LFAST memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	12.6.1/203
5	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	12.6.2/205
9	LFAST Correlator Control Register (LFAST_COCR)	32	R/W	0000_000Eh	12.6.3/206
D	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	12.6.4/207
11	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	12.6.5/209
15	LFAST Rate Change Delay Control Register (LFAST_RCDCR)	32	R/W	000F_0000h	12.6.6/209
19	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	12.6.7/210
21	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	12.6.8/211
3D	LFAST PLL Control Register (LFAST_PLLCR)	32	R/W	0000_005Ch	12.6.9/212
41	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	12.6.10/215
45	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	12.6.11/218

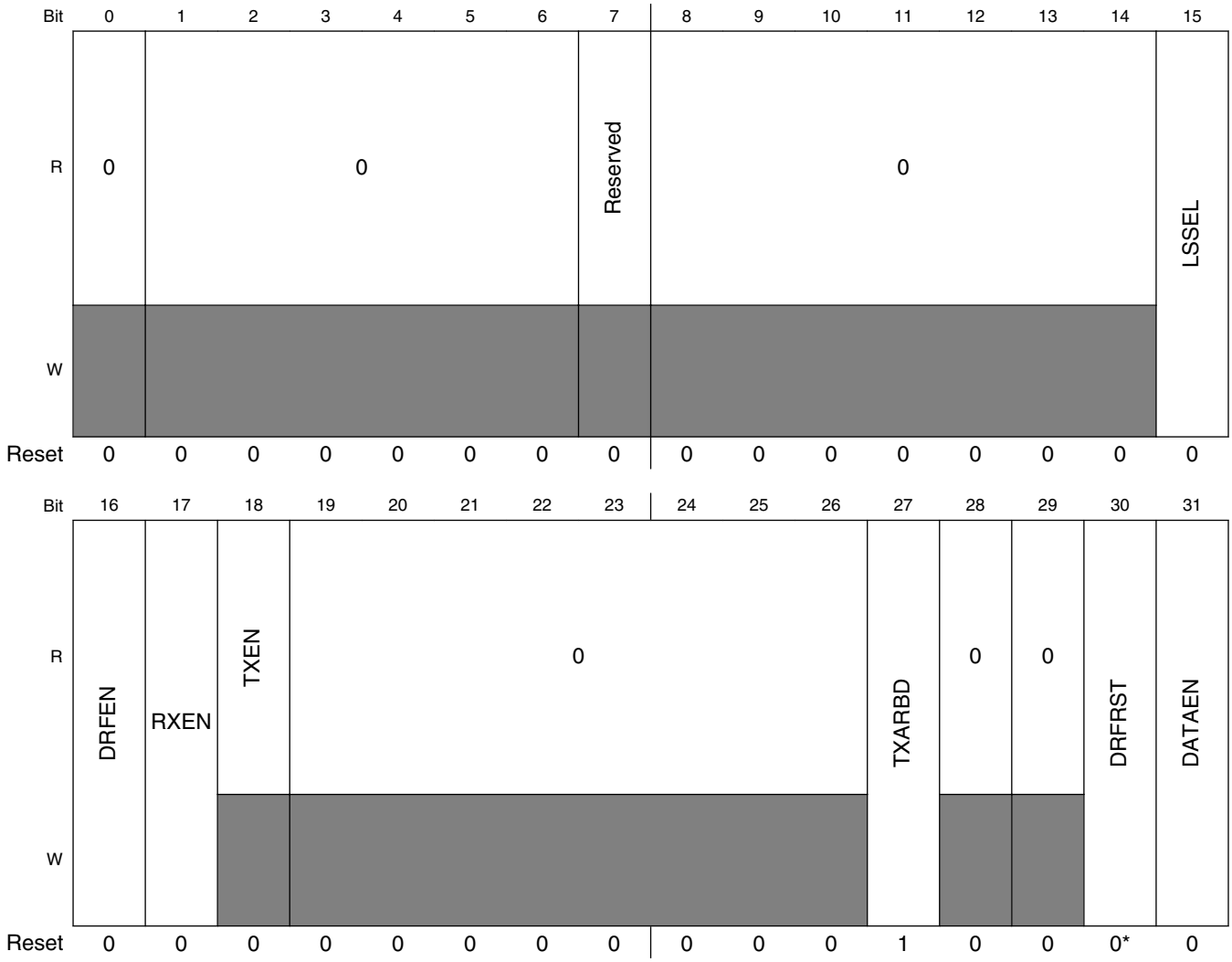
Table continues on the next page...

LFAST memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
49	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	12.6.12/218
4D	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	12.6.12/218
51	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	12.6.12/218
55	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	12.6.12/218
59	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	12.6.12/218
5D	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	12.6.12/218
61	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	12.6.12/218
65	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	12.6.12/218
69	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	12.6.12/218
81	LFAST Global Status Register (LFAST_GSR)	32	R	See section	12.6.13/219
95	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	12.6.14/220
99	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	12.6.15/221
9D	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	12.6.16/223
A1	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	R/W	0000_0000h	12.6.17/225
A5	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	12.6.18/227
A9	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	12.6.19/228
AD	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	12.6.20/229
B1	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	12.6.20/229
B5	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	12.6.20/229
B9	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	12.6.20/229
BD	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	12.6.20/229
C1	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	12.6.20/229
C5	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	12.6.20/229
C9	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	12.6.20/229
CD	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	12.6.20/229

12.6.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 1h base + 0h offset = 1h



* Notes:
- DRFRST field: Set by user software and then cleared by system hardware.

LFAST_MCR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved.

Table continues on the next page...

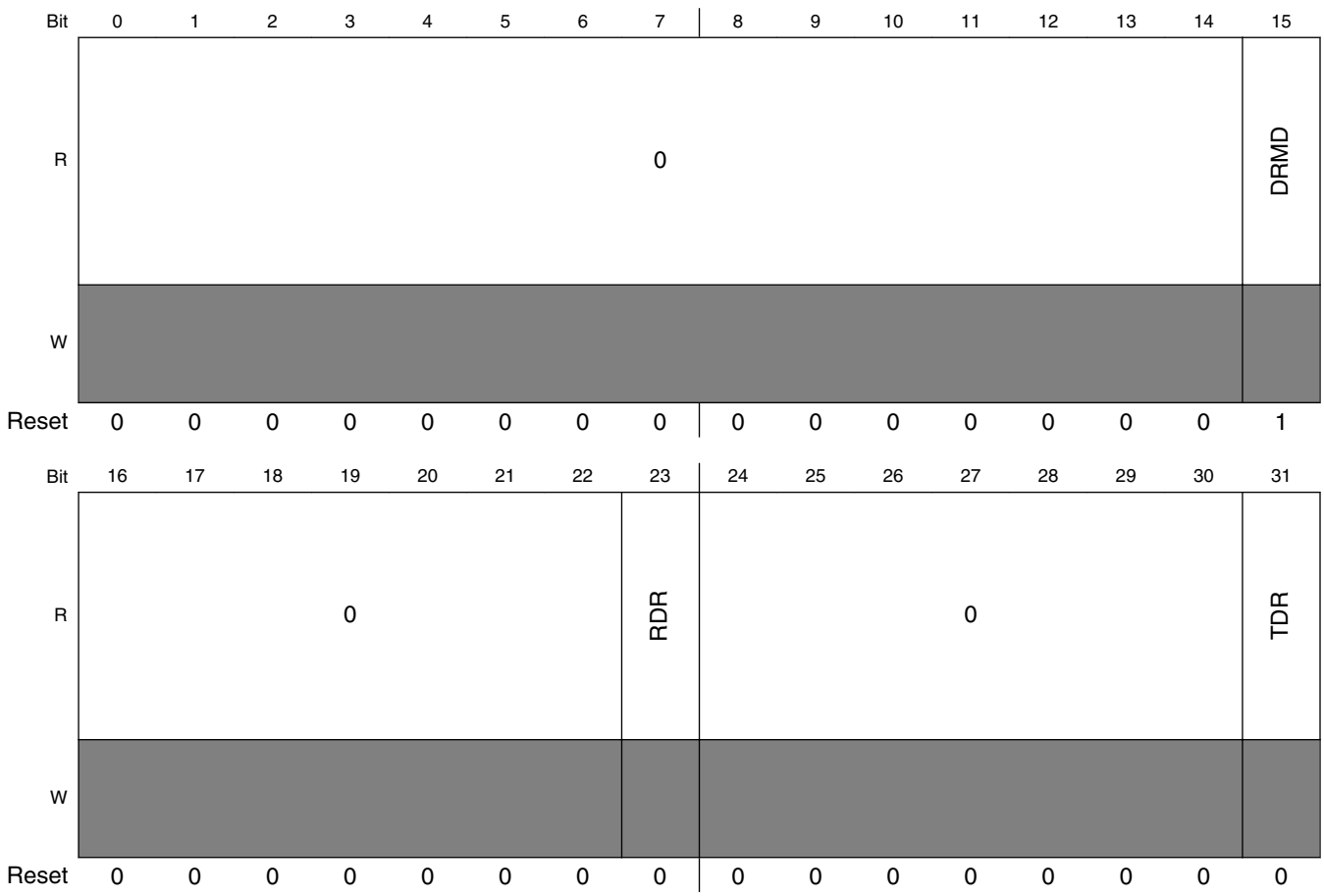
LFAST_MCR field descriptions (continued)

Field	Description
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit shows the whether the transmitter is enabled or disabled. This field is read only in slave only mode (LFAST_GSR[DUALMD] = 0) and must not be written. 0 LFAST transmitter Interface is disabled. 1 LFAST transmitter Interface is enabled.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset.. 0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

12.6.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 1h base + 4h offset = 5h



LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. This bit shouldn't be modified by S/W and left as SCR[DRMD] =1.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit shows the data rate of the receiver. This bit should not be modified by S/W in slave only mode (LFAST_GSR[DUALMD] = 0). 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is 312/320 Mbps.

Table continues on the next page...

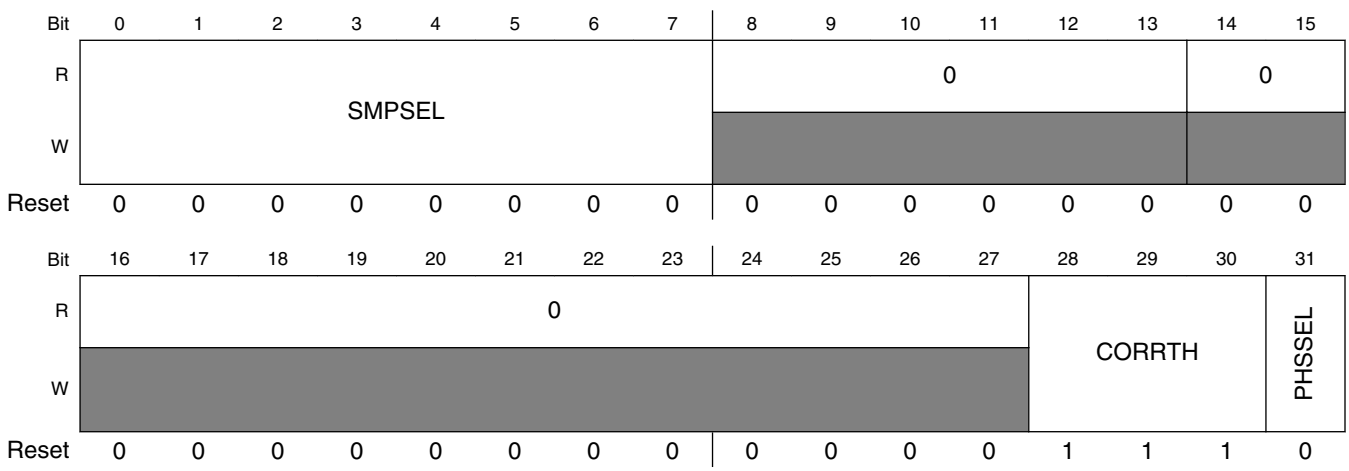
LFAST_SCR field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit shows the data rate of the transmitter. This bit should not be modified by S/W in slave only mode (LFAST_GSR[DUALMD] = 0). 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is 312 /320 Mbps.

12.6.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 1h base + 8h offset = 9h



LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0. 00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected

Table continues on the next page...

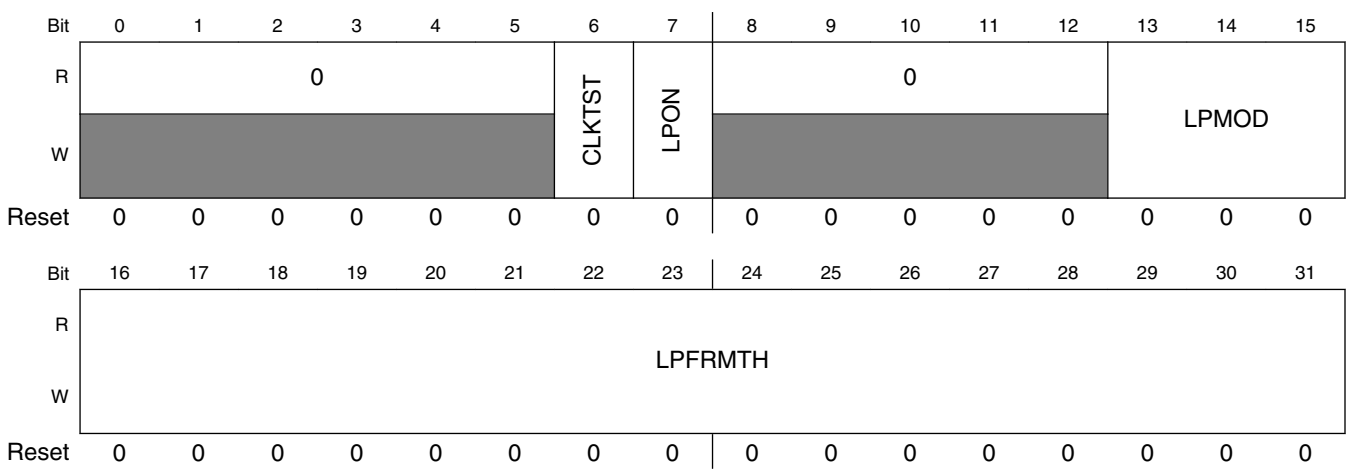
LFAST_COCR field descriptions (continued)

Field	Description
	40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

12.6.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 1h base + Ch offset = Dh

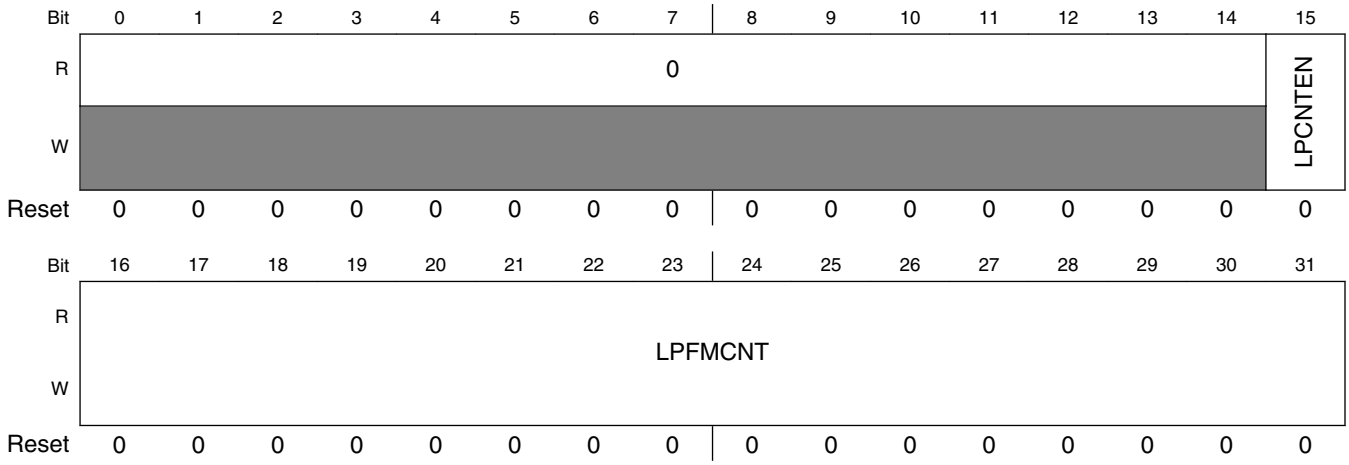


LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by H/W on reception of an ICLC command. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved 110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

12.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 1h base + 10h offset = 11h

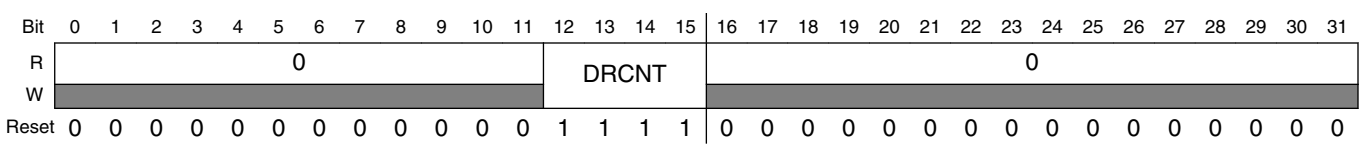


LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload ... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

12.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 1h base + 14h offset = 15h



LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 1h base + 18h offset = 19h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT			
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ... 12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles

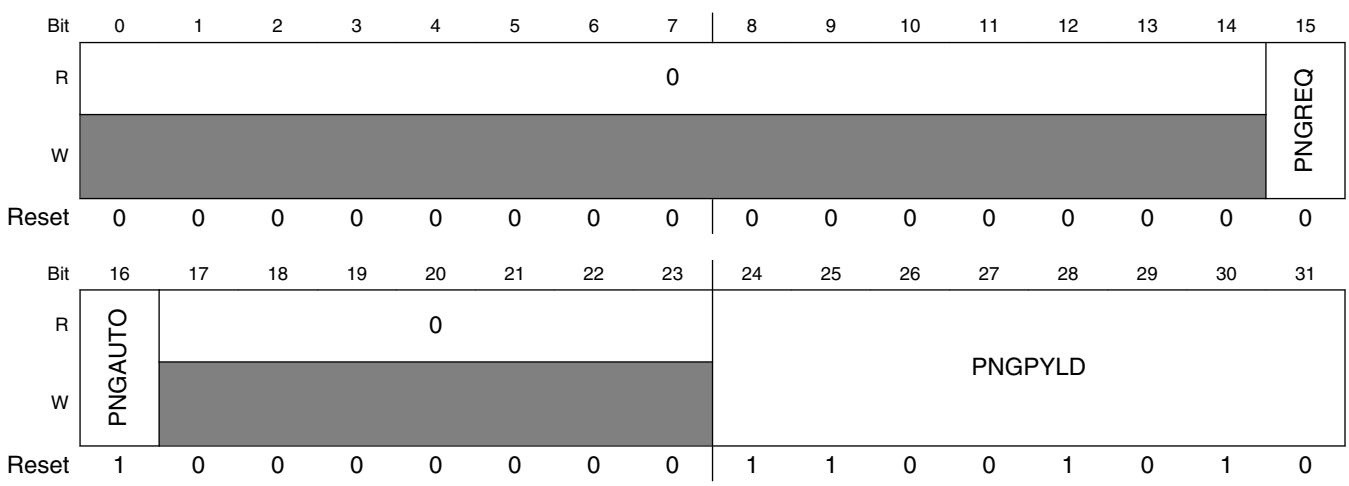
Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 µs) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 µs) ... Eh 14 cycles Fh 15 cycles

12.6.8 LFAST Ping Control Register (LFAST_PICR)

Address: 1h base + 20h offset = 21h



LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

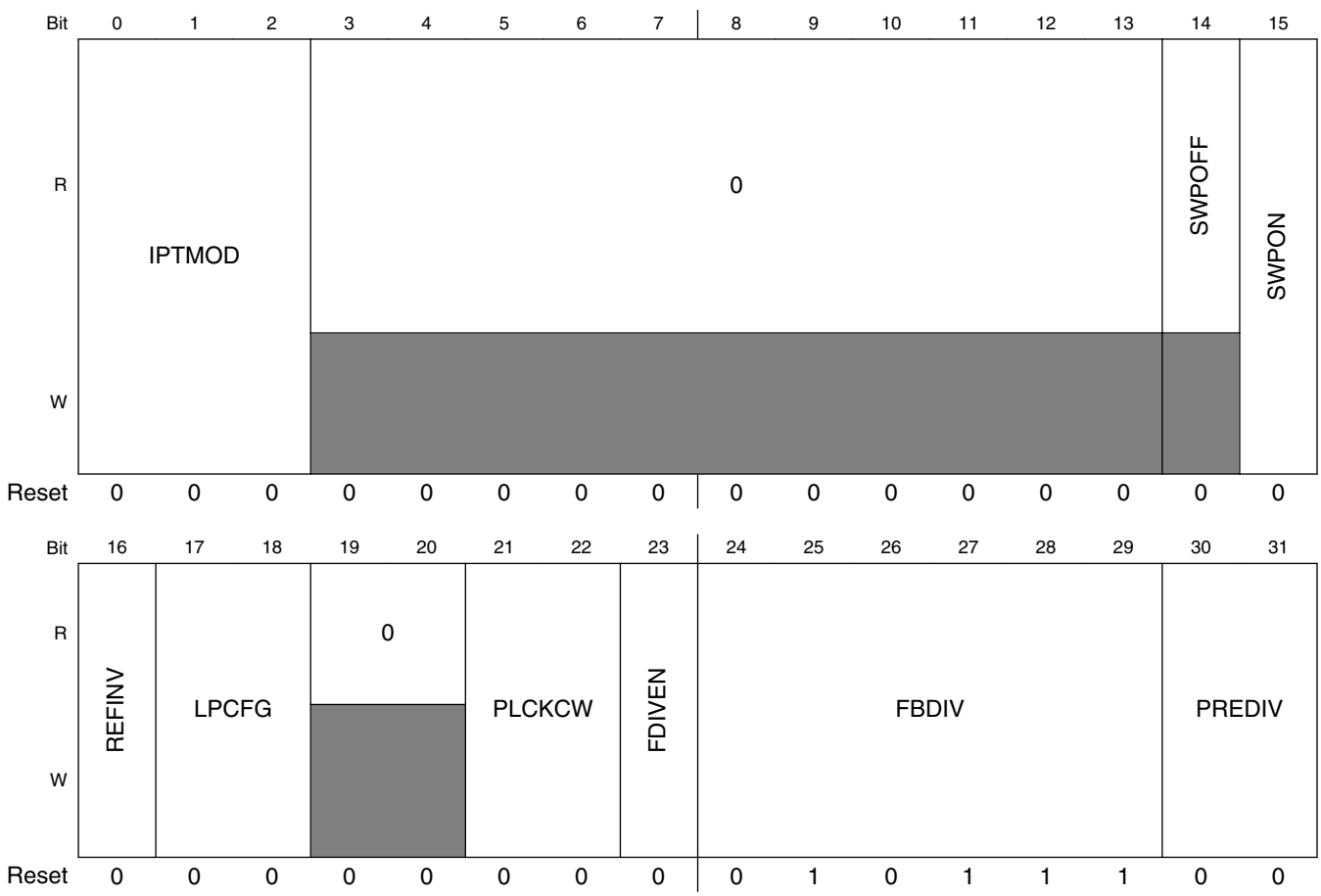
Table continues on the next page...

LFAST_PICR field descriptions (continued)

Field	Description
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. This bit should not be modified by S/W.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0.

12.6.9 LFAST PLL Control Register (LFAST_PLLCR)

Address: 1h base + 3Ch offset = 3Dh



LFAST_PLLCR field descriptions

Field	Description
0–2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
3–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SWPOFF	This bit shouldn't be modified by S/W.
15 SWPON	This bit shouldn't be modified by S/W.
16 REFINV	Inverts reference clock edge to PFD. If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Do not inverted 1 Invert
17–18 LPCFG	PLL Loop Optimization for sysclk frequency 00 1 × IBASE current 01 0.5 × IBASE current 10 1.5 × IBASE current 11 2 × IBASE current
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24–29 FBDIV	Feedback Division factor for VCO output clock. This field can only be written when LFAST_MCR[DRFEN] = 0. 00h No clock output 01h – 0Ah Reserved

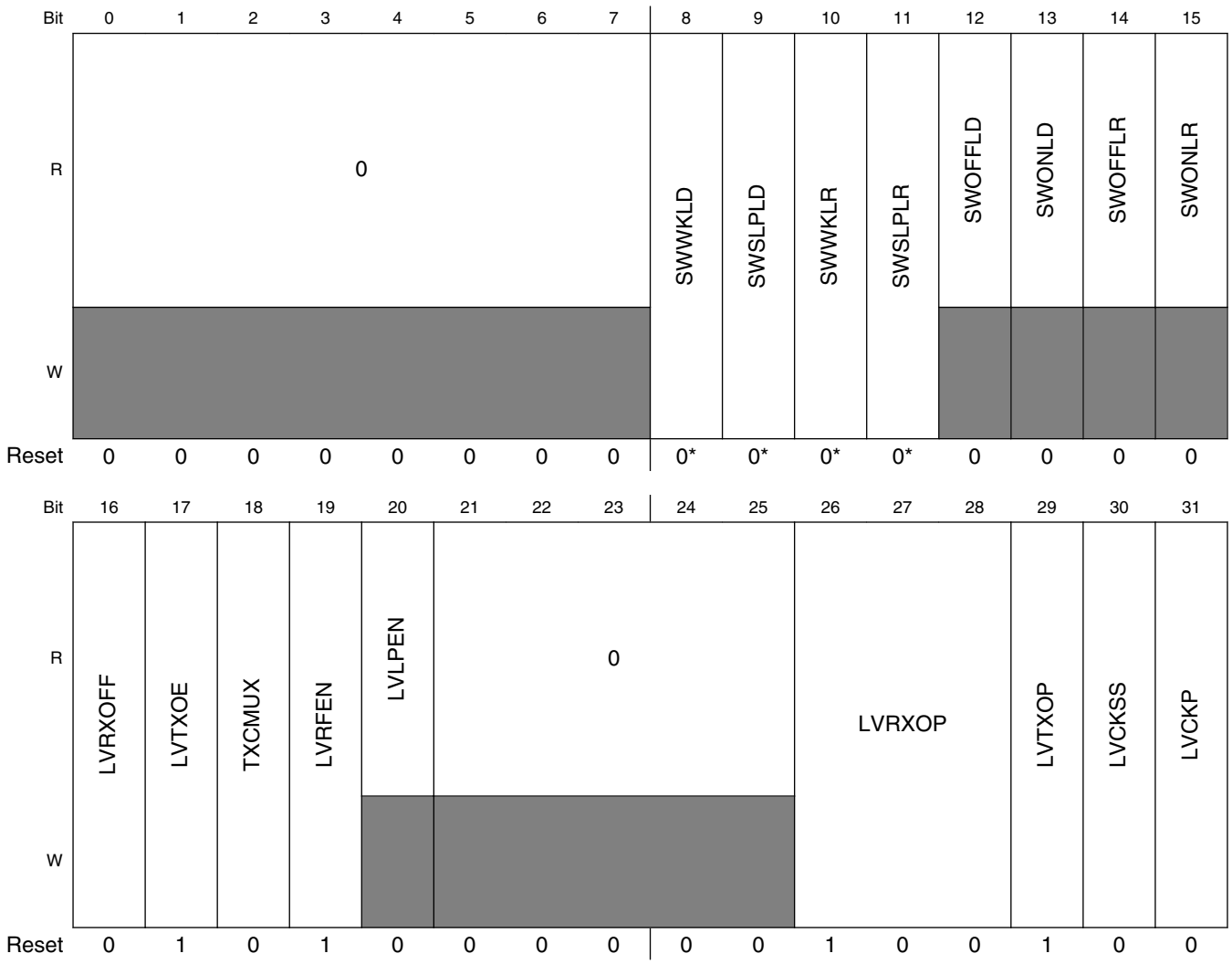
Table continues on the next page...

LFAST_PLLCR field descriptions (continued)

Field	Description
	0Bh Divide by 12 (11.5, if FDIVEN = 1) ... ... 1Fh Divide by 32 (31.5, if FDIVEN = 1) 20h – 3Fh Reserved
30–31 PREDIV	Division factor for PLL Reference Clock input. This field can only be written when LFAST_MCR[DRFEN] = 0. 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

12.6.10 LFAST LVDS Control Register (LFAST_LCR)

Address: 1h base + 40h offset = 41h



* Notes:
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
13 SWONLD	SW signal to turn ON the LVDS LD. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
15 SWONLR	SW signal to turn ON the LVDS LR. This bit can be changed only when DRFEN is high. This bit shouldn't be modified by S/W.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled

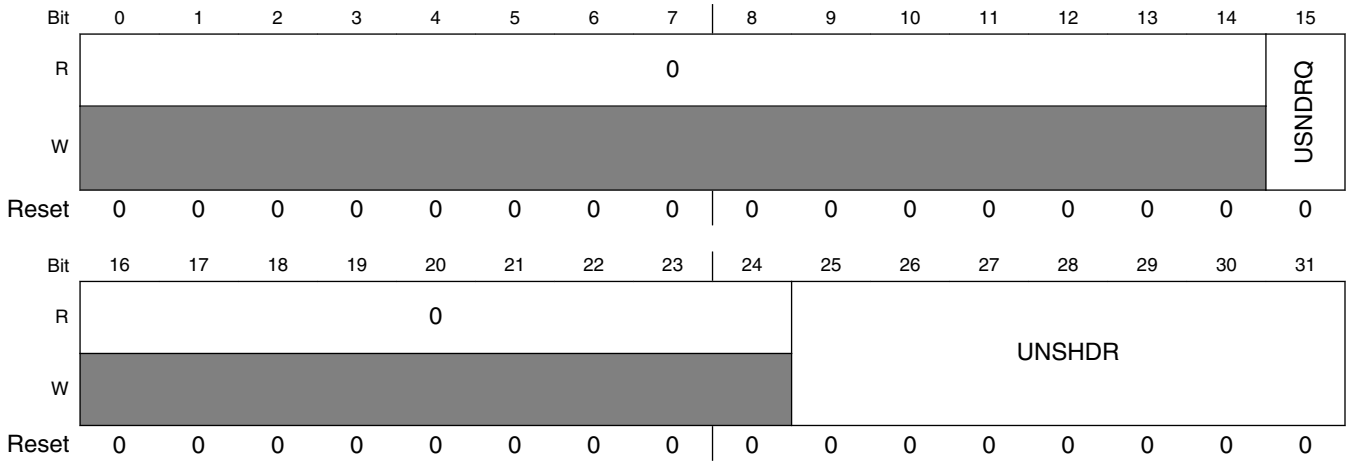
Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
	1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable The bit is reflected by output signal ipp_digrf_lvds_lpbk_en.The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–28 LVRXOP	LVRXOP[2] – Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination LVRXOP[1] – Reserved LVRXOP[0] – Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than 320Mbps. 1 Required for LFAST receiver maximum baud rate (320Mbps).
29 LVTXOP	Control signal for LFASTand Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Data select. This bit is used for selecting use of clock sampled data or normal data in LVDS. 0 normal data to be used inside the LVDS 1 synchronized data to be used inside the LVDS
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

12.6.11 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 1h base + 44h offset = 45h

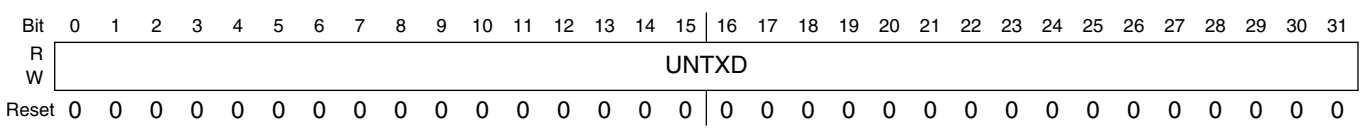


LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

12.6.12 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)

Address: 1h base + 48h offset + (4d × i), where i=0d to 8d

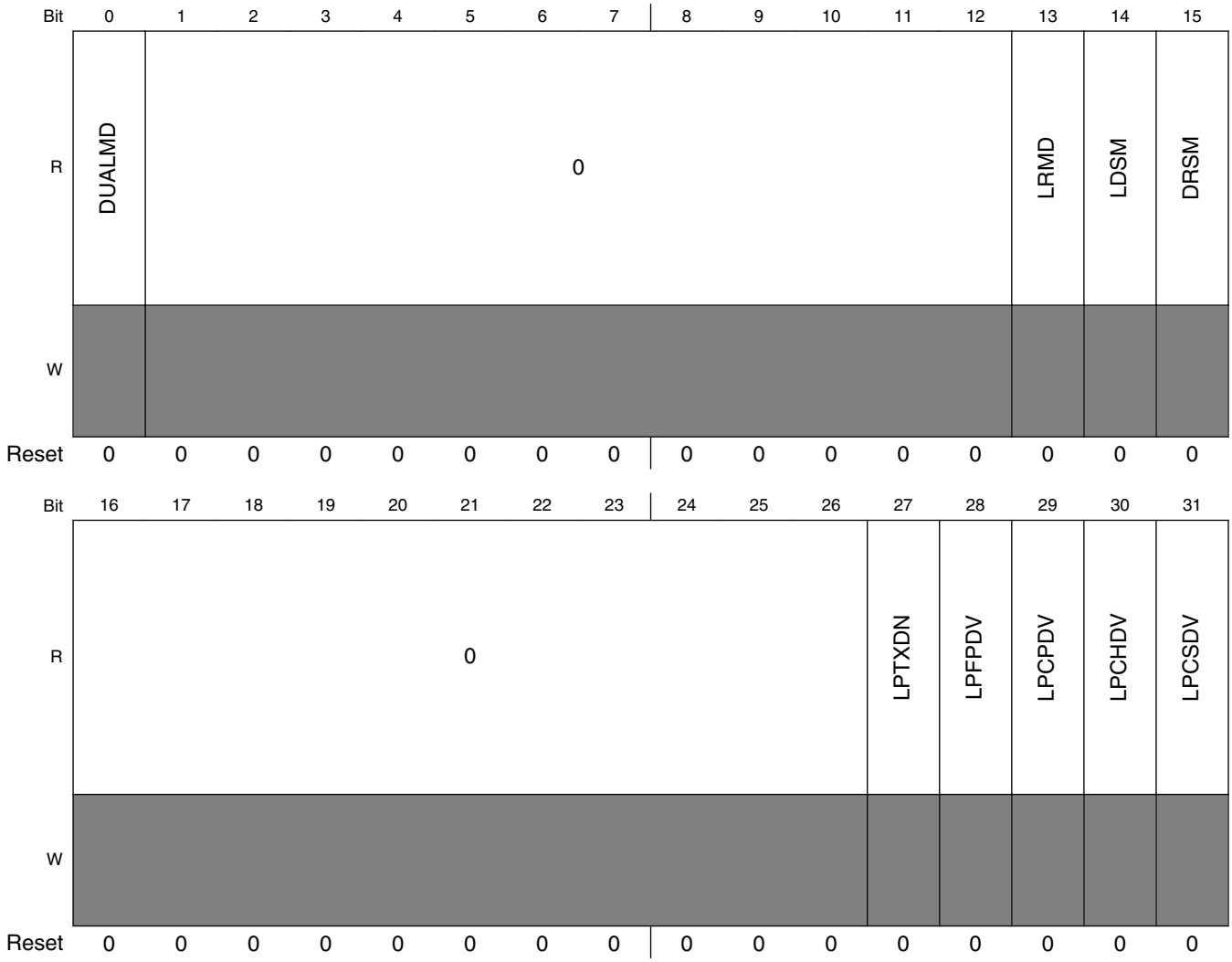


LFAST_UNSTDR*n* field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

12.6.13 LFAST Global Status Register (LFAST_GSR)

Address: 1h base + 80h offset = 81h



LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active.

Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

12.6.14 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 1h base + 94h offset = 95h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		RXDCNT						0					RXFCNT			0		TXDCNT							0					TXFCNT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

12.6.15 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 1h base + 98h offset = 99h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

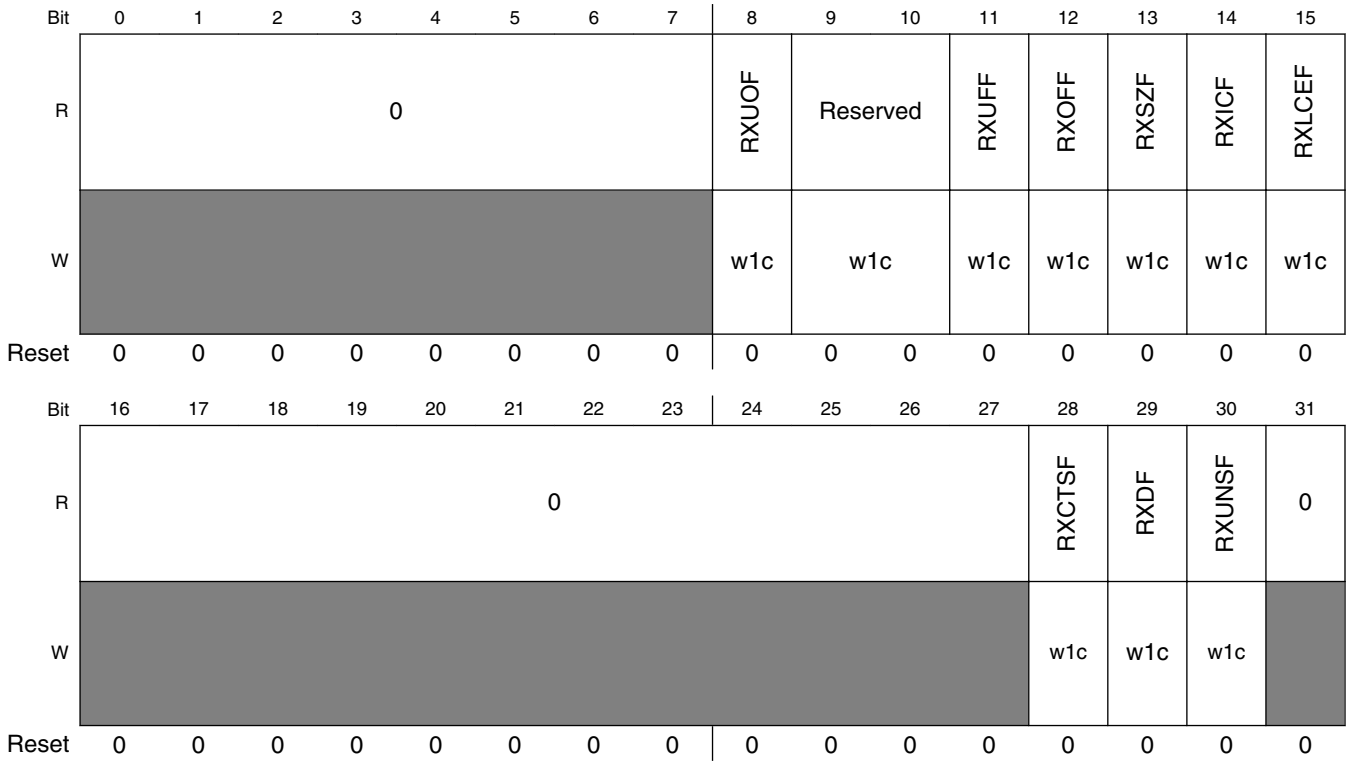
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

12.6.16 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 1h base + 9Ch offset = 9Dh



LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9–10 Reserved	Reserved. The User should ignore the bit in this mode. This field is reserved.
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.6.17 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)

Address: 1h base + A0h offset = A1h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received

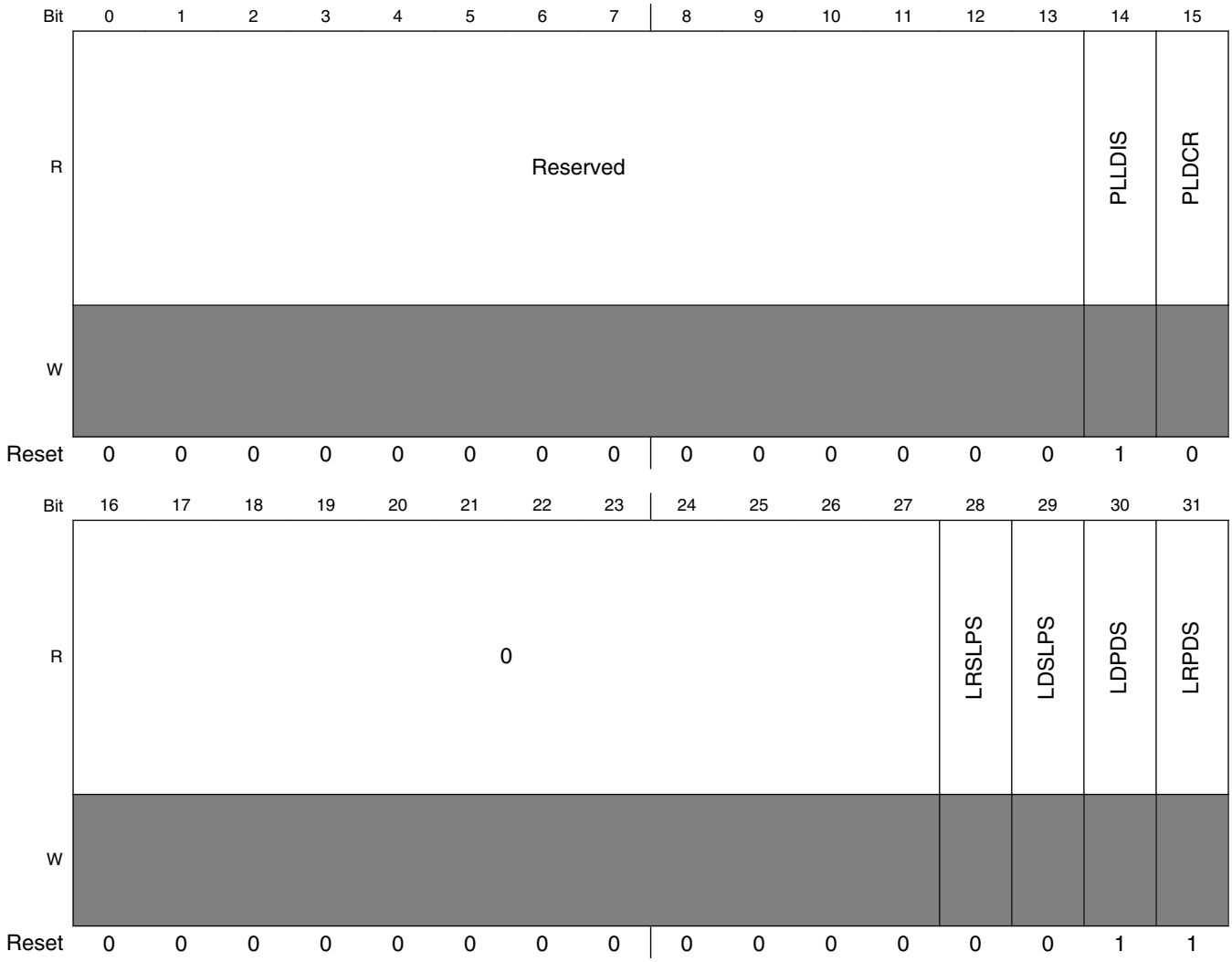
Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
	0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

12.6.18 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 1h base + A4h offset = A5h



LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

12.6.19 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 1h base + A8h offset = A9h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0					URPCNT		
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

12.6.20 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)

Address: 1h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR*n* field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

12.7 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements[6]. All registers of the LFAST module are classified as non-safety relevant.

12.8 Functional description

6. See the “Functional Safety” chapter for register classification details.

12.8.1 LFAST Interface enable signal (lfast_sysclk_en)

The LFAST interface enable signal (lfast_sysclk_en) signal is present only in the case LFAST Slave . It also functions as the LFAST Interface control Signal. The interface control functions are:

LFAST interface enable (lfast_sysclk_en) negation:

1. If LCR[SWONLD] = 0, the LD LVDS pad is shut down.
2. If LCR[SWONLR] = 0 the LR LVDS pad is shut down.
3. The LFAST slave exits the test mode (for example, loopback mode or clock test mode). Then S/W writes MCR[CLKTST] = 0 and MCR[LPON] = 0.
4. The LFAST slave Rx and Tx interfaces are set to Low Speed mode. Then write SCR[TDR] = 0 and SCR[RDR] = 0.
5. The Tx and Rx interface controllers are disabled.
6. The status registers are cleared.

LFAST interface enable (lfast_sysclk_en) assertion:

1. The LD LVDS pad is power up, if all conditions are met:
   - LCR[SWOFFLD] = 0.
   - LCR[SWONLD] = 1.
   - MCR[TXEN] = 1.
   - When frame ICR[ICLCPLD] = 31h is received.
2. LR LVDS pad is powered up if all conditions are met:
   - LCR[SWOFFLR] = 0.
   - LCR[SWONLR] = 1.
   - MCR[RXEN] = 1.

12.8.2 Line Receiver

This section describes the Line Receiver.

12.8.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 12-4 shows the block diagram of Uplink Controller.

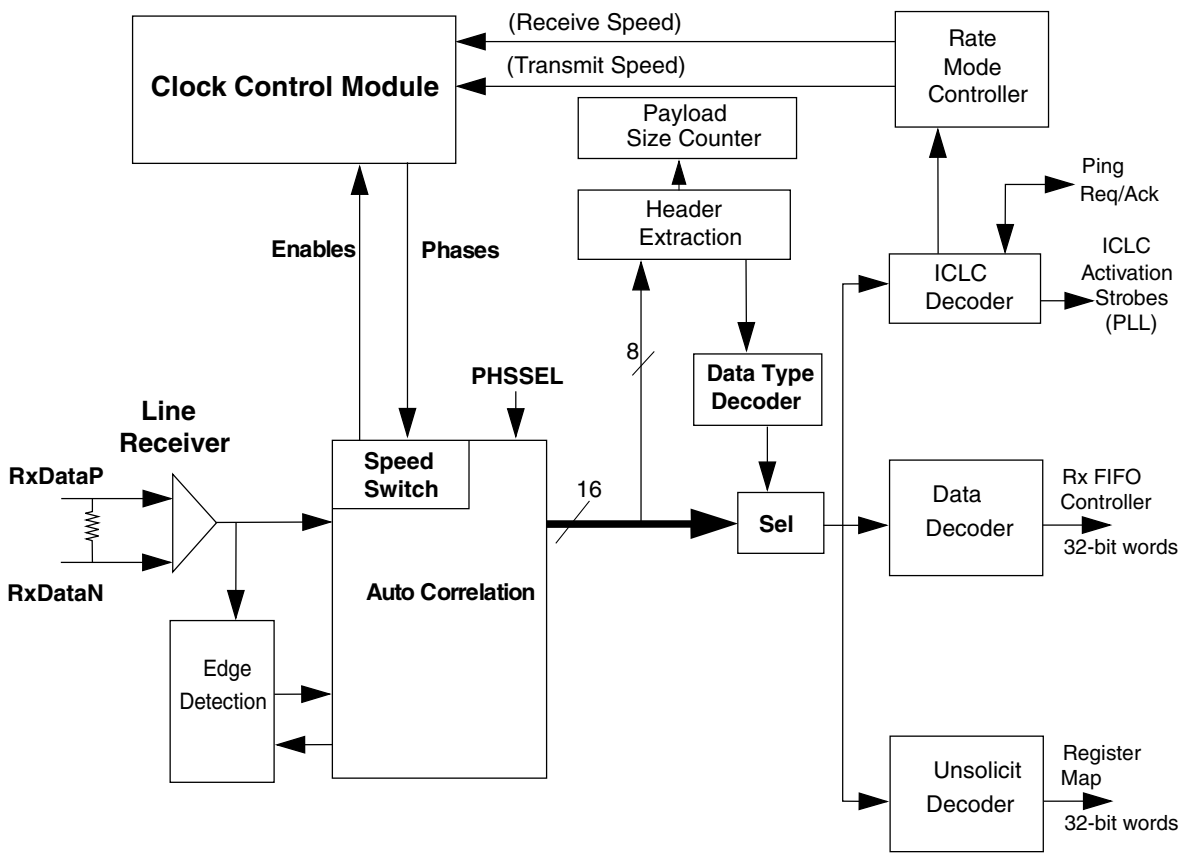


Figure 12-4. Top Level Receive Controller

12.8.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

12.8.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

12.8.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 12-5 shows the block diagram of Hunt Correlation mode.

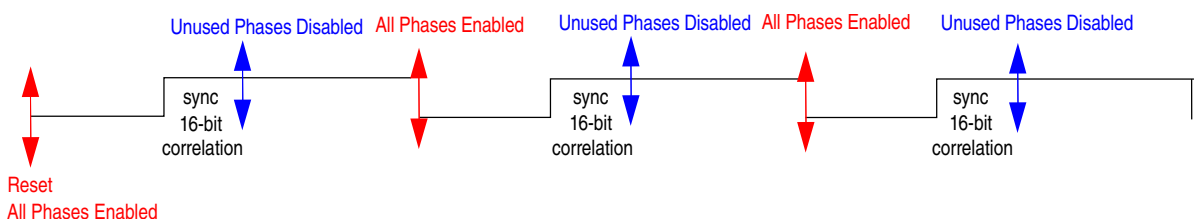


Figure 12-5. Hunt Correlation mode

12.8.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

12.8.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

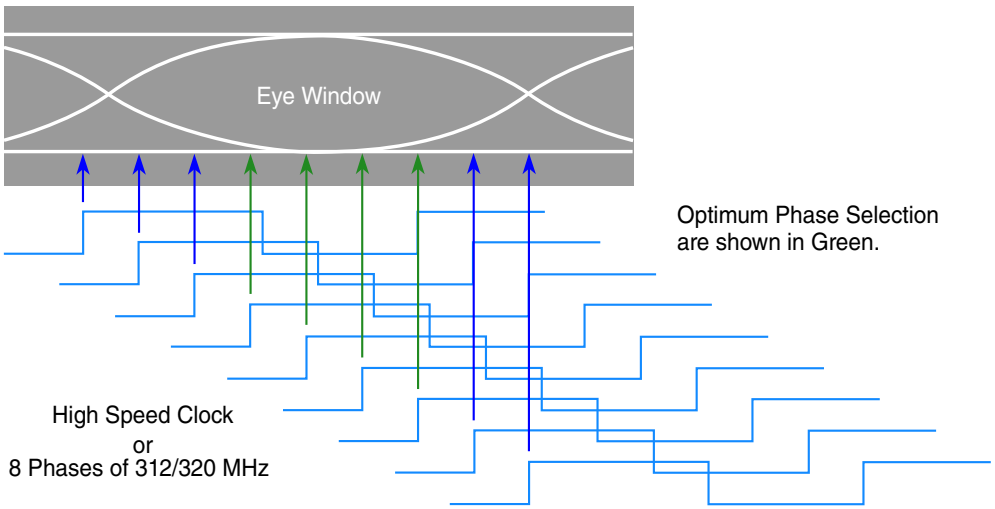


Figure 12-6. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

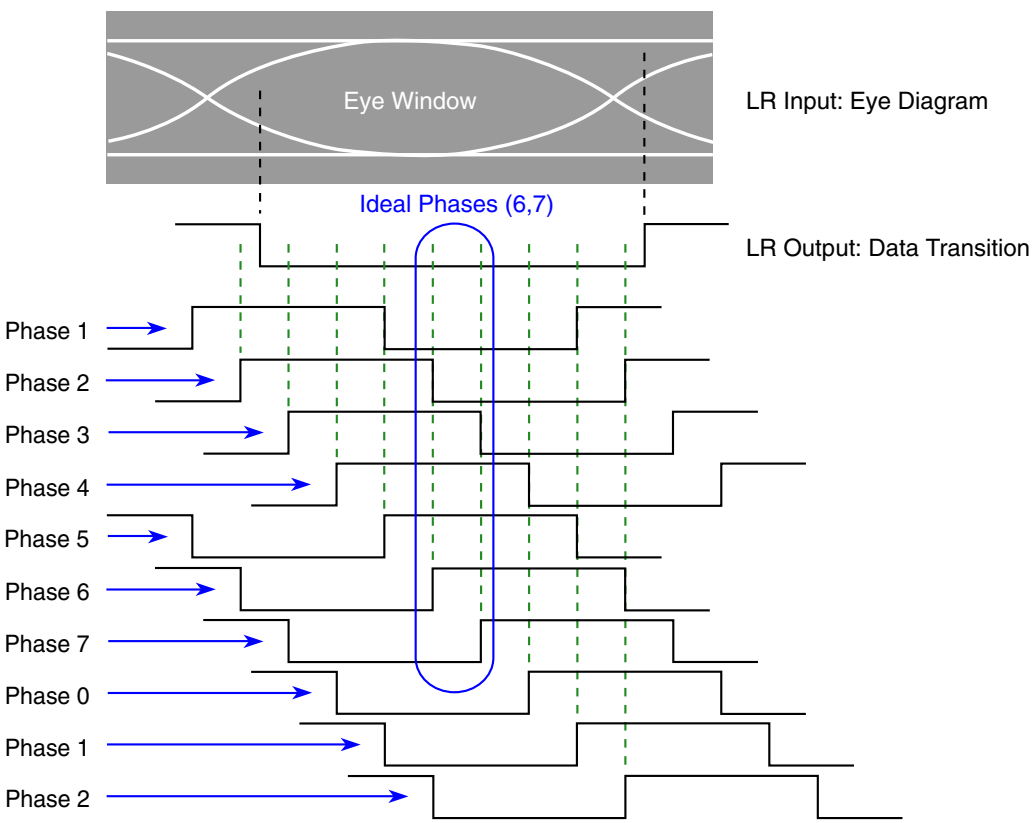


Figure 12-7. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in Figure 12-8 and Figure 12-9.

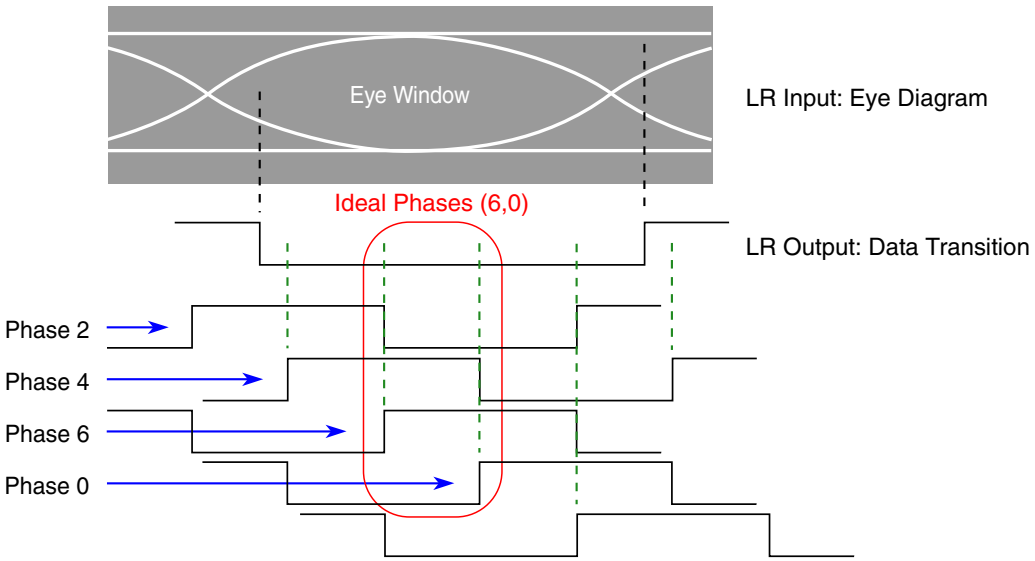


Figure 12-8. High Speed 4 Phase Clock Alignment Example

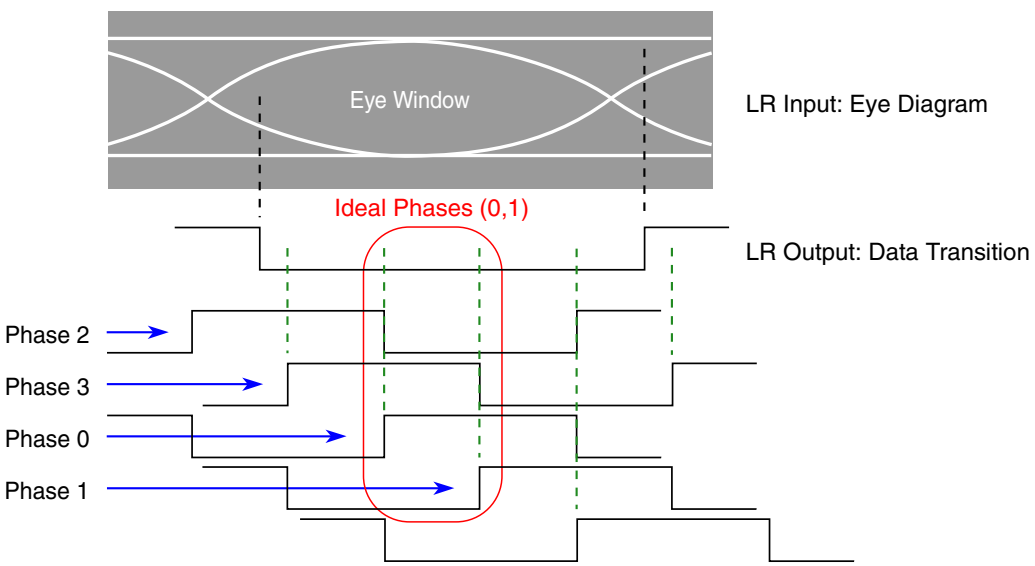


Figure 12-9. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed (312/320 MHz) 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in Table 12-3, Table 12-4, and Table 12-5 before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 12-3. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 12-4. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See Table 12-4, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 12-5. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 12-5. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See Table 12-5, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

12.8.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

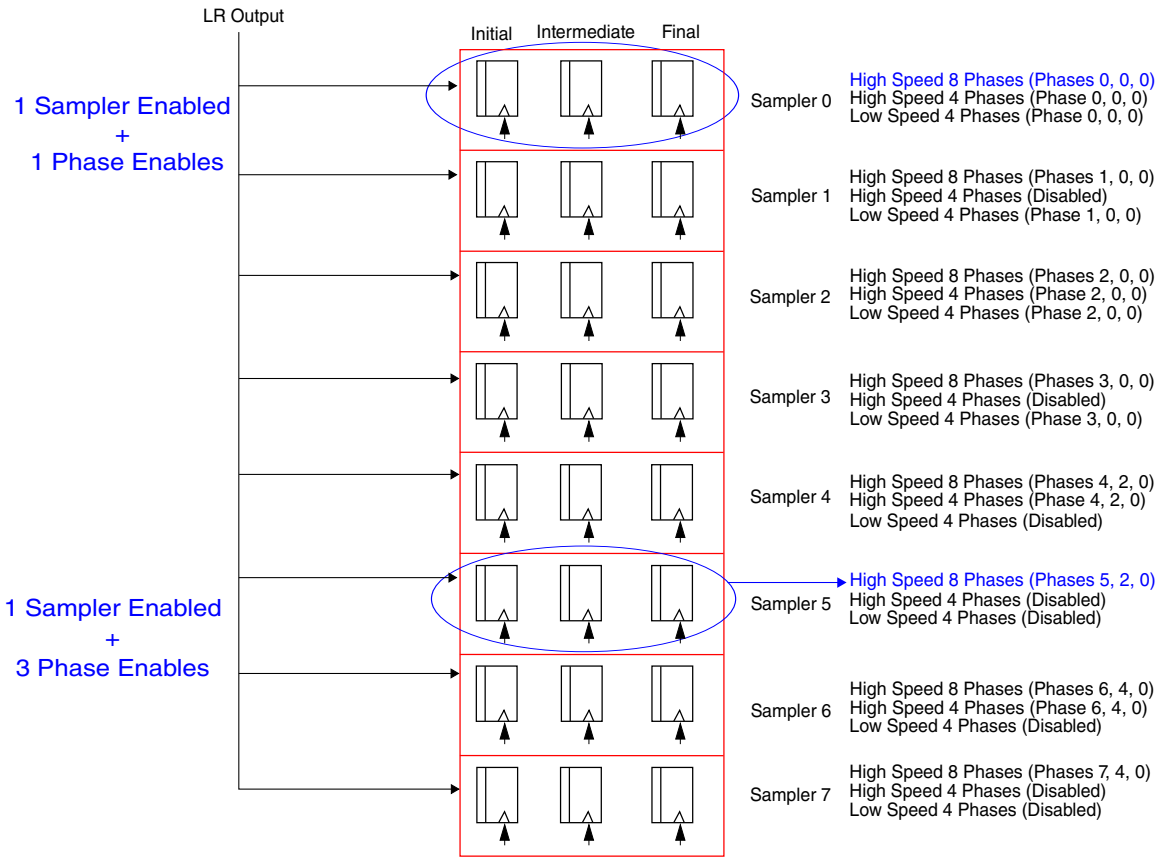


Figure 12-10. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

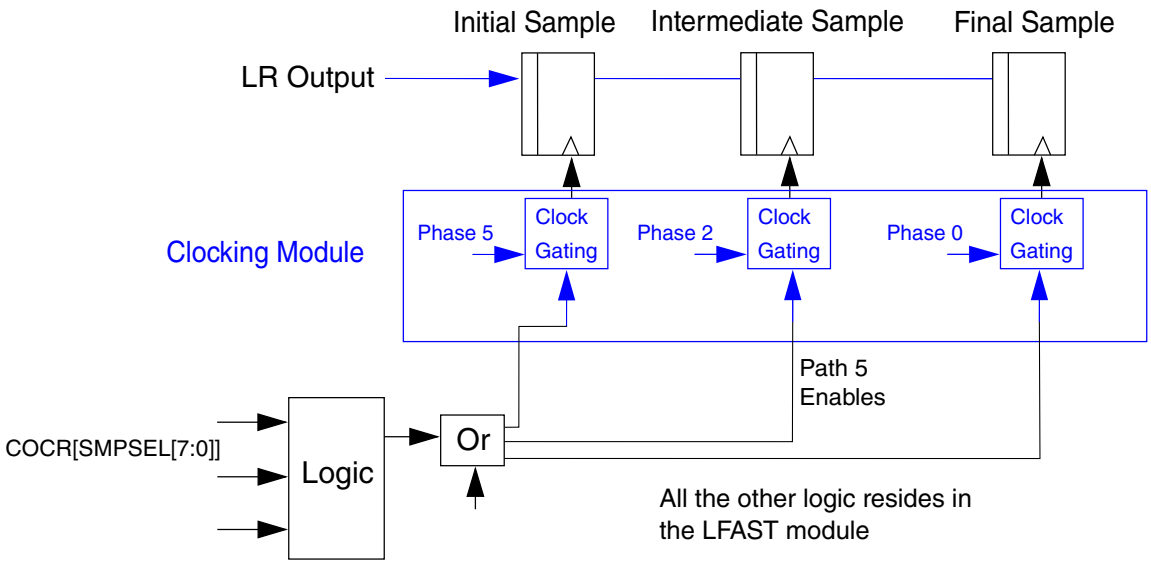


Figure 12-11. Sampler 5 Logic

12.8.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the Figure 12-12.

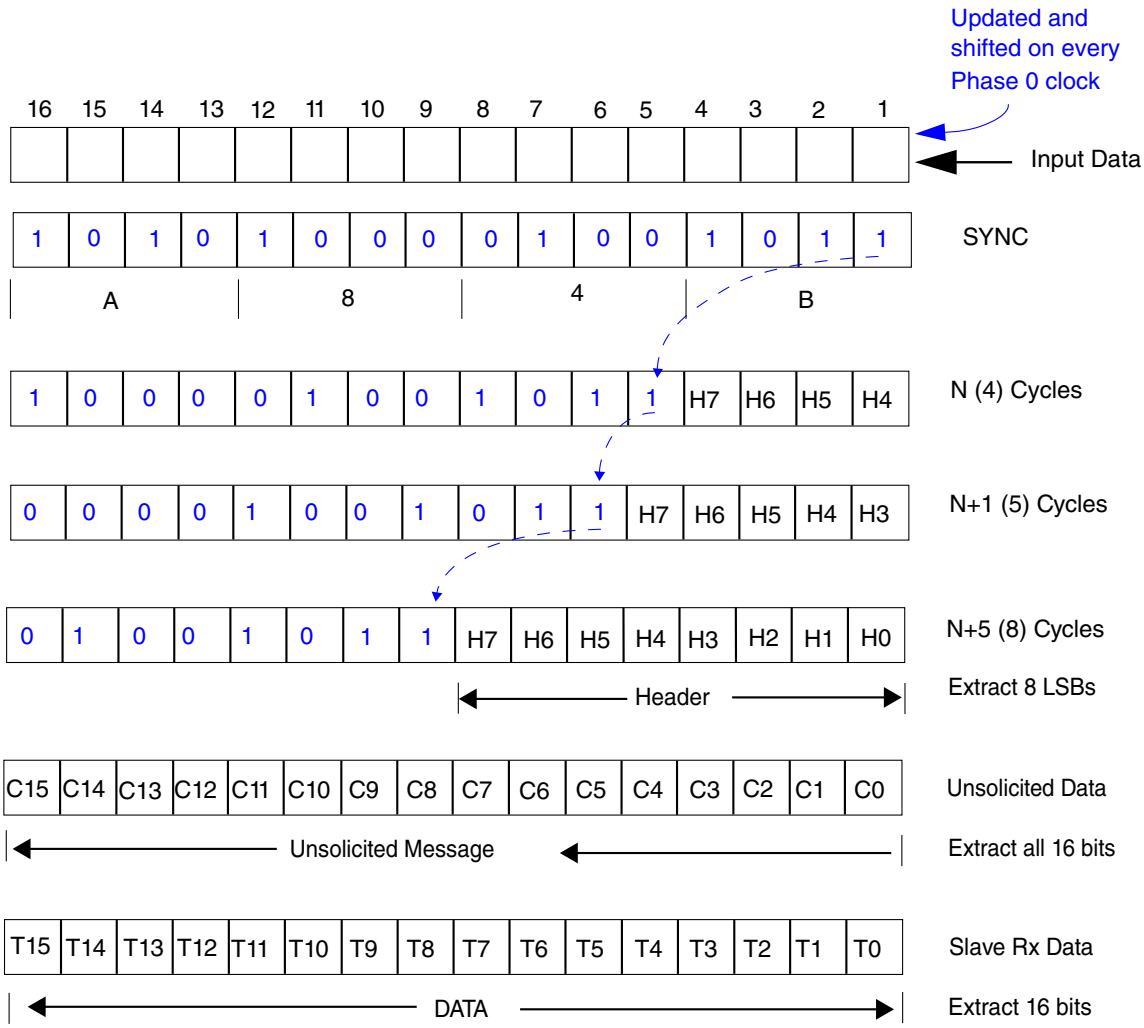


Figure 12-12. Extracting Header and Payload from the 16-bit output from Auto correlation.

12.8.3 Transmit Controller

This section describes the Transmit Controller.

12.8.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

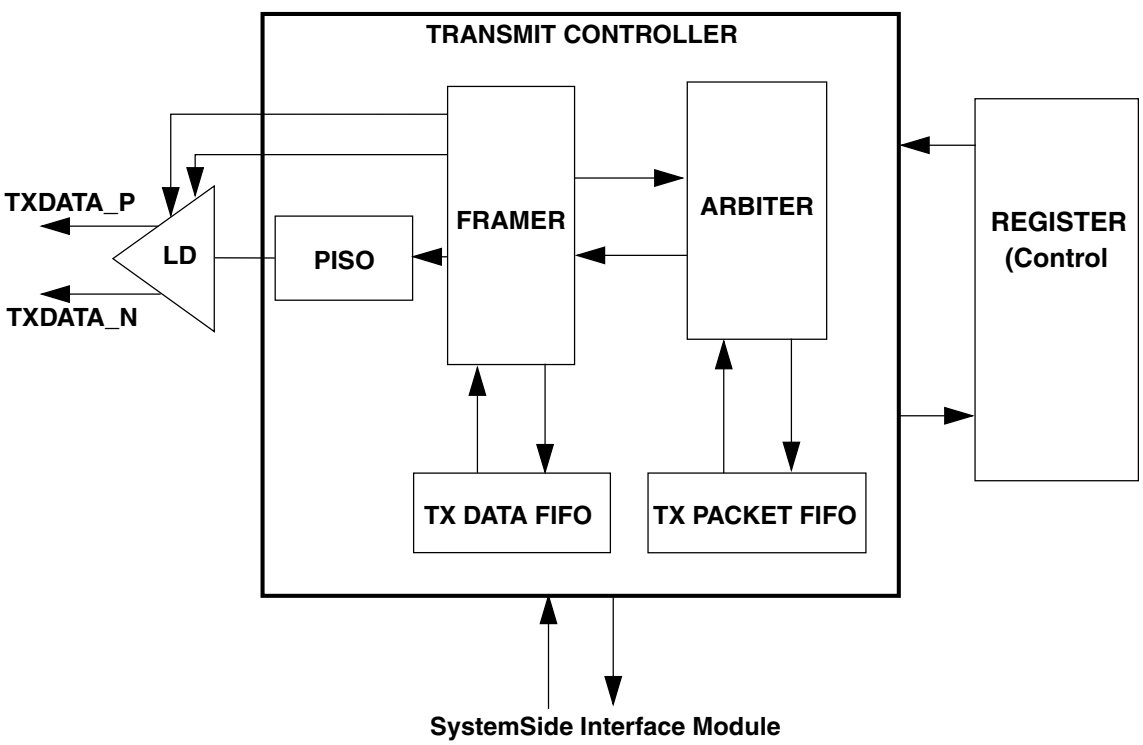


Figure 12-13. Transmit Controller Connections

12.8.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller
- LFAST Slave: LFAST interface enable (lfast_sysclk_en)

The level of priority for each request is shown in Table 12-6

Table 12-6. Priority Levels for the Tranmsit Controller

Request	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
LFAST interface enable (lfast_sysclk_en)	1	• LFAST interface enable (lfast_sysclk_en) is asserted • LFAST interface enable (lfast_sysclk_en) is negated
Tx Interface disabled	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	• TMCR[LPON] = 1 • LFAST Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
Ping response request	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	• Valid only if Last Frame with header b0 = 1 received from the peer LFAST • UNSTCR[USNDRQ] = 1

Table continues on the next page...

Table 12-6. Priority Levels for the Tranmsit Controller (continued)

Request	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
Data frame from Tx FIFO	7	• Tx FIFO contains one or more frames • Last Frame with header b0 = 1 received from the peer LFAST device • MCR[DATAEN] = 1
Clock mode test	8	• TMCR[CLKTST] = 0 • LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

12.8.3.3 Line Driver digital connections

12.8.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

  The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down

completely. In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

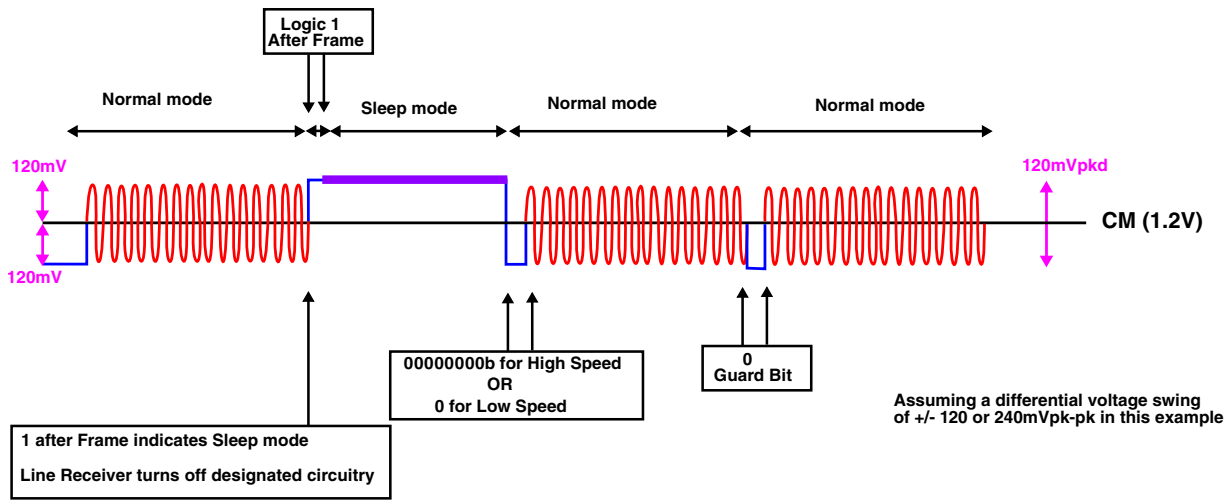


Figure 12-14. Example of Sleep mode

As shown in Figure 12-14 before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 12-7. Line Driver States

LD State	LFAST Slave Triggers
Shutdown	• LFAST interface enable (lfast_sysclk_en) negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions

12.8.4 Frames supported

Table 12-8 provides the frames supported and their permissible payload sizes

Table 12-8. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	Rx – 0100b,1000b – 1011b Tx – All	YES	YES	TX: 96 RX:128
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO[1]	8
CTS Frame	n/a	NO	NO	n/a
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

12.8.5 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

12.8.5.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

12.8.5.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and

schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

12.8.5.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer

12.8.5.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded.

12.8.5.2 Unsolicited flow

12.8.5.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

12.8.5.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
    - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
        - UNSTCR[USNDRQ] = 0
        - TISR[TXUNSF] = 1
    - If UNSTCR[USNDRQ] = 0 then:
        - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
        - Program the unsolicited frame header in UNSTCR[UNSHDR].
        - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.

12.8.5.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.

12.8.5.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

12.8.5.3.1 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

12.8.5.3.1.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

12.8.5.3.1.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

12.8.5.3.1.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

12.8.5.3.1.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

12.8.5.3.1.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

12.8.5.3.1.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

12.8.5.3.1.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable (lfast_sysclk_en).

12.8.6 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

12.8.6.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 12-9. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

12.8.6.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. Bit b0 of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

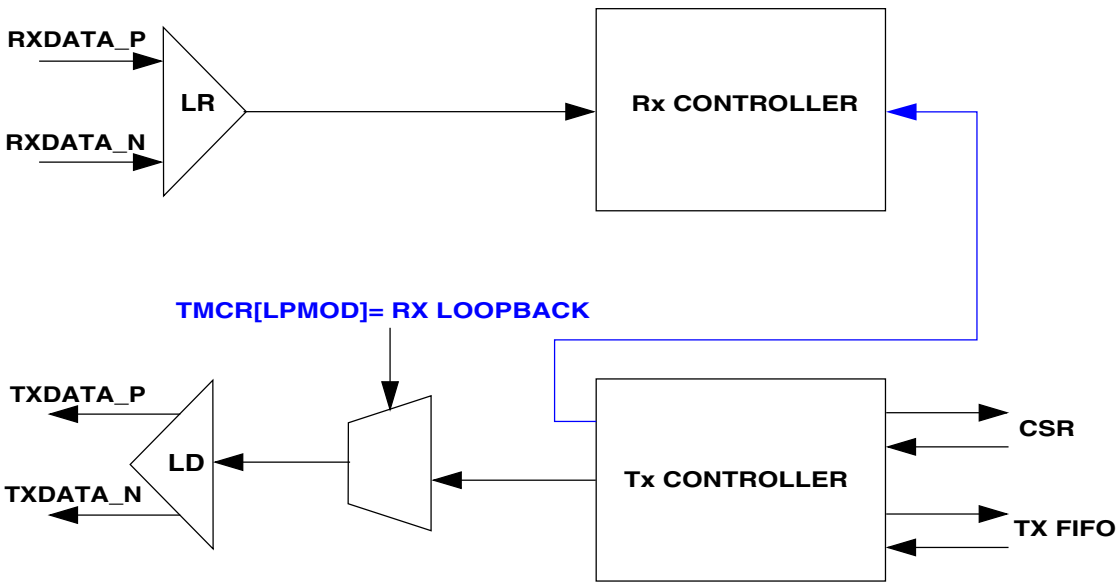


Figure 12-15. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master
- For LFAST Slave: - Deassertion of the LFAST interface enable (lfast_sysclk_en).

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

12.8.6.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

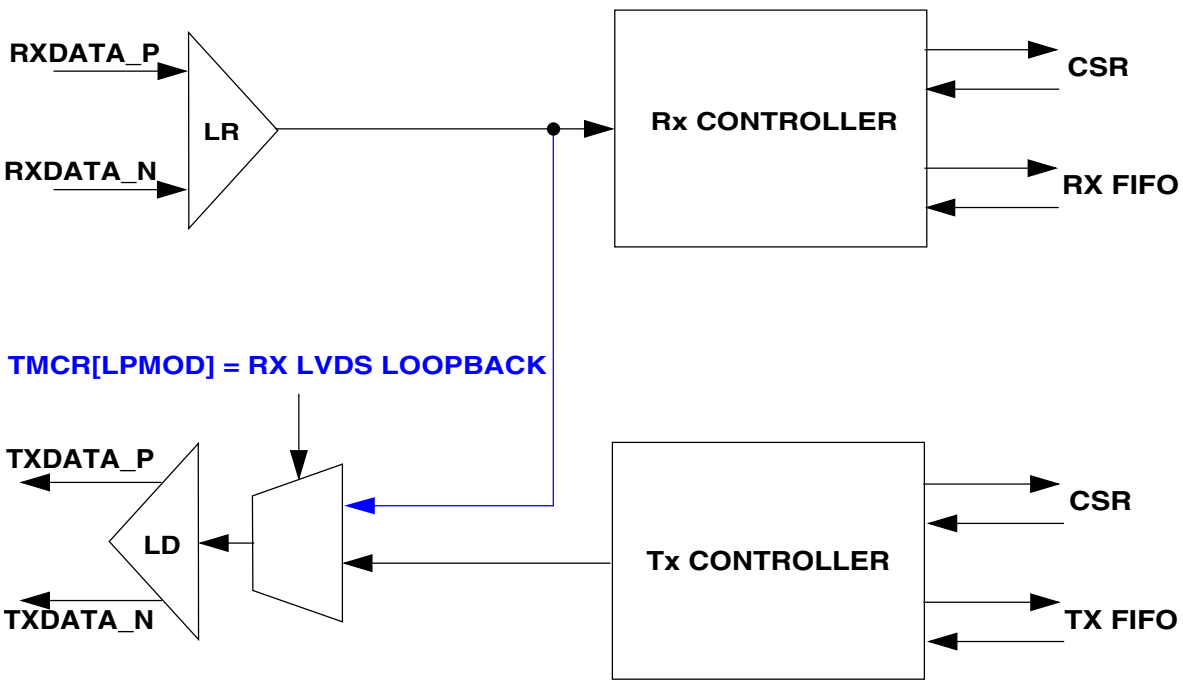


Figure 12-16. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
   - S/W writes TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.
  - Deassertion of the LFAST interface enable (lfast_sysclk_en).

12.8.6.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

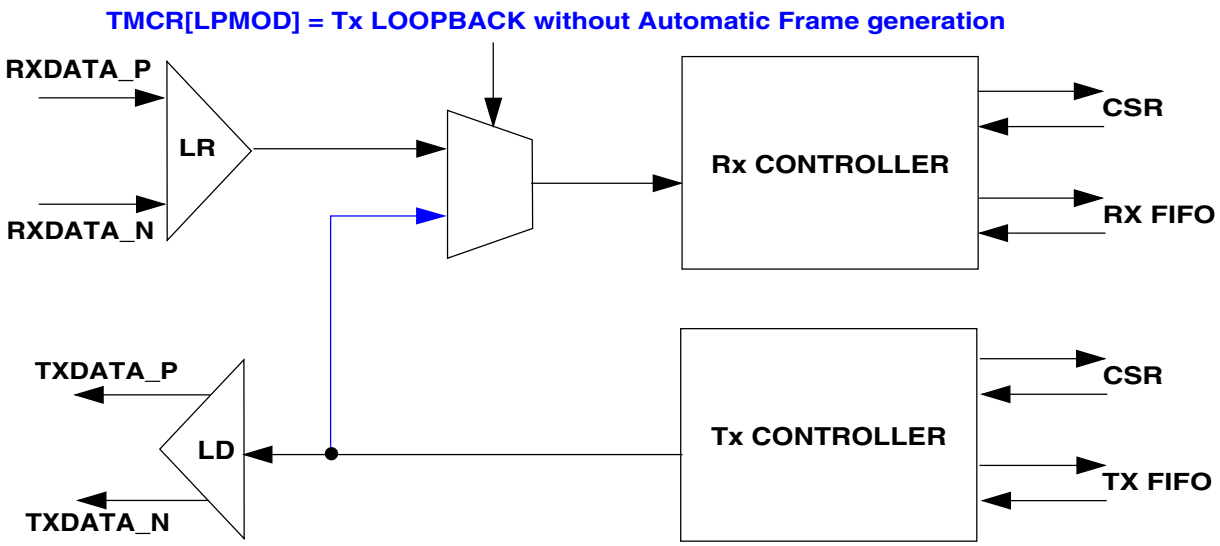


Figure 12-17. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

All frames and error flags are decoded in this mode.

12.8.6.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

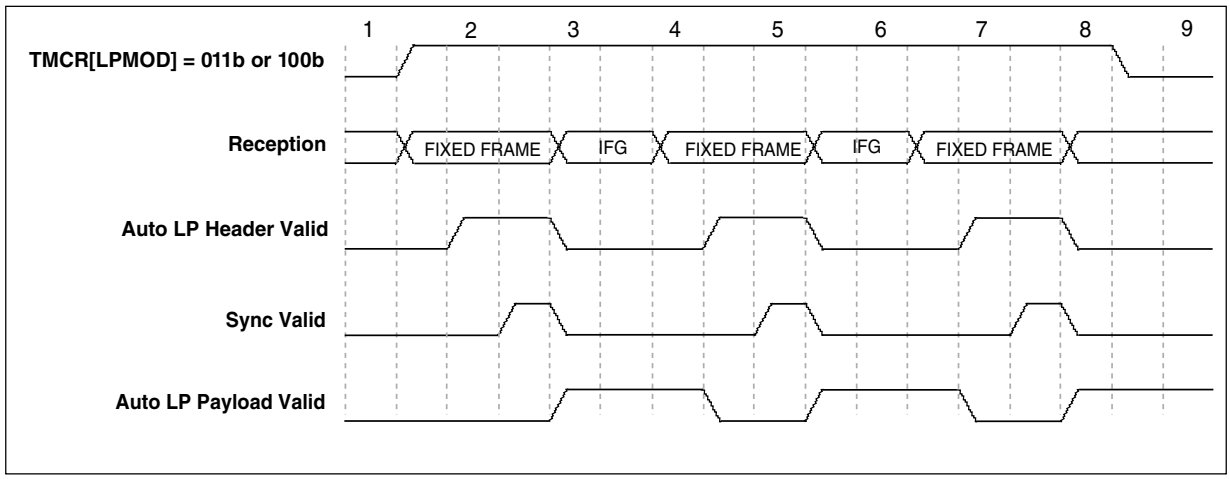


Figure 12-18. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

12.8.6.1.4.1 Tx loopback mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

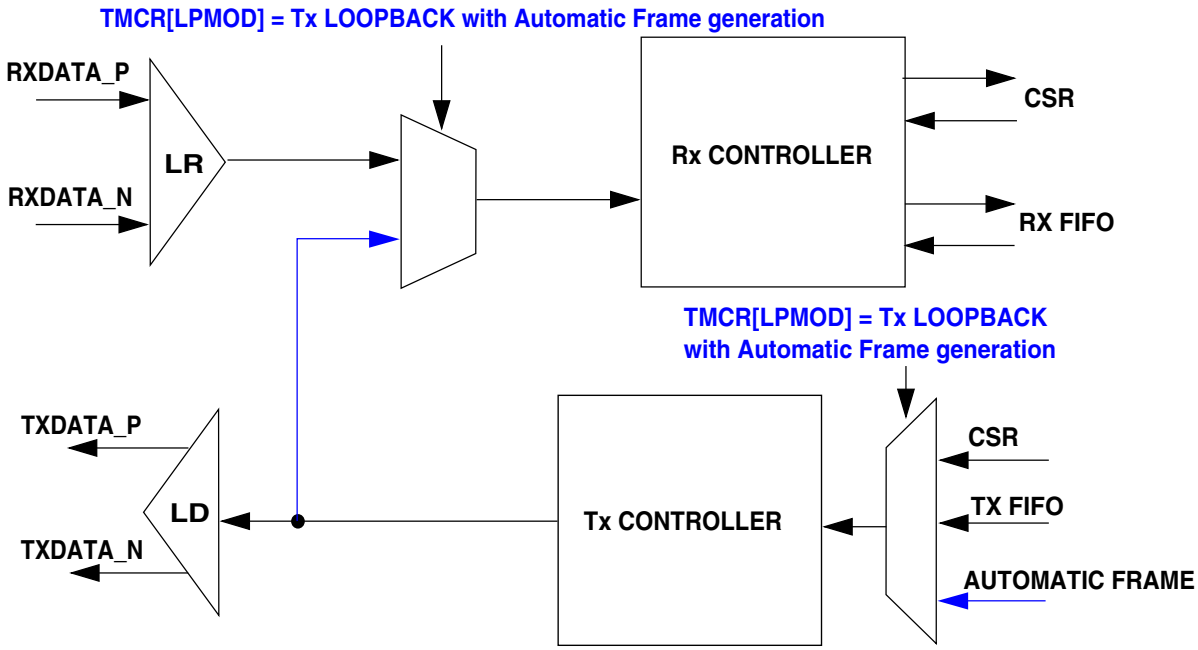


Figure 12-19. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

12.8.6.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

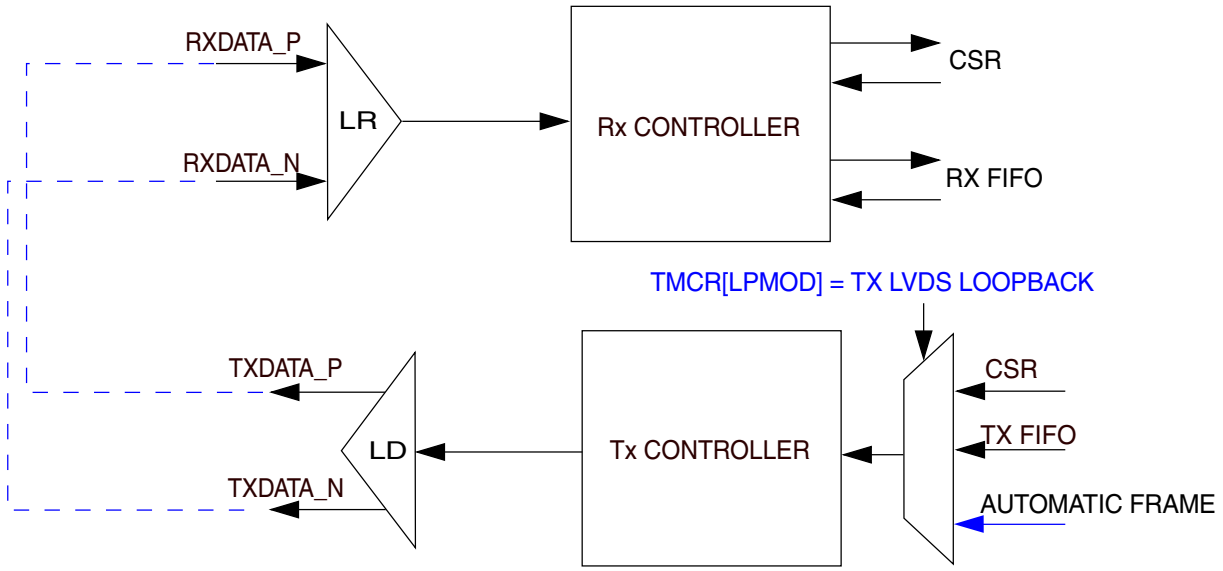


Figure 12-20. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

12.8.6.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 34h (Clock Test mode on) from LFAST Master

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

12.9 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 12-10. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	6 × 32 max 2 packets	8 × 32 max 2 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	N/A	1 × 8 max 1 packet[3]	Registers PICR[PNGPYLD]
CTS Frame	N/A	N/A	N/A

3. Ping response packet

12.10 Resets

The various blocks in LFAST are reset as described in the table below.

Table 12-11. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]
LFAST interface enable (lfast_sysclk_en)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers except the following: TISR, RISR, RIISR, GSR[LPTXDN] and GSR[LPFPDV] These bits remain reseted while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: Negedge only • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON]

Table continues on the next page...

Table 12-11. Recommended receive exception handling mechanism (continued)

Reset	Blocks Reset
	• ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ] These bits doesn't remain reseted while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: positive edge only • TISR, • RISR, • RIISR, • GSR[LPTXDN] and GSR[LPFPDV] • These bits doesn't remain reseted while the LFAST interface enable (lfast_sysclk_en) is High.

12.11 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

12.11.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

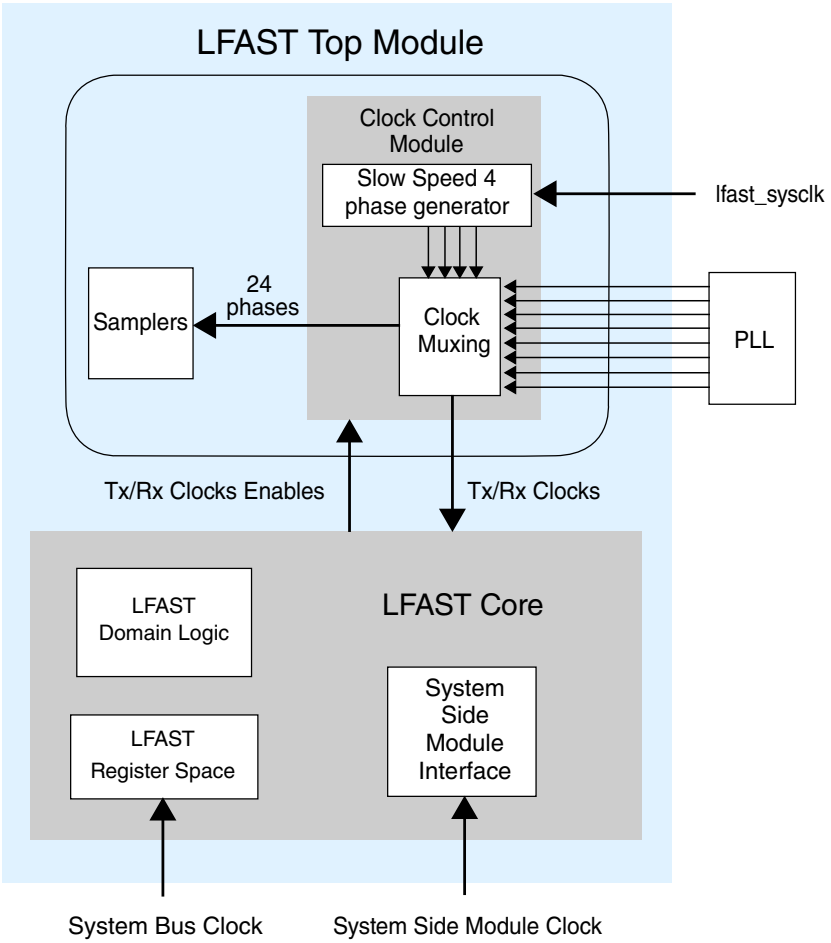


Figure 12-21. LFAST module clock domains

12.11.2 Slow speed clock

12.11.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in Figure 12-22, Figure 12-23 and Figure 12-24.

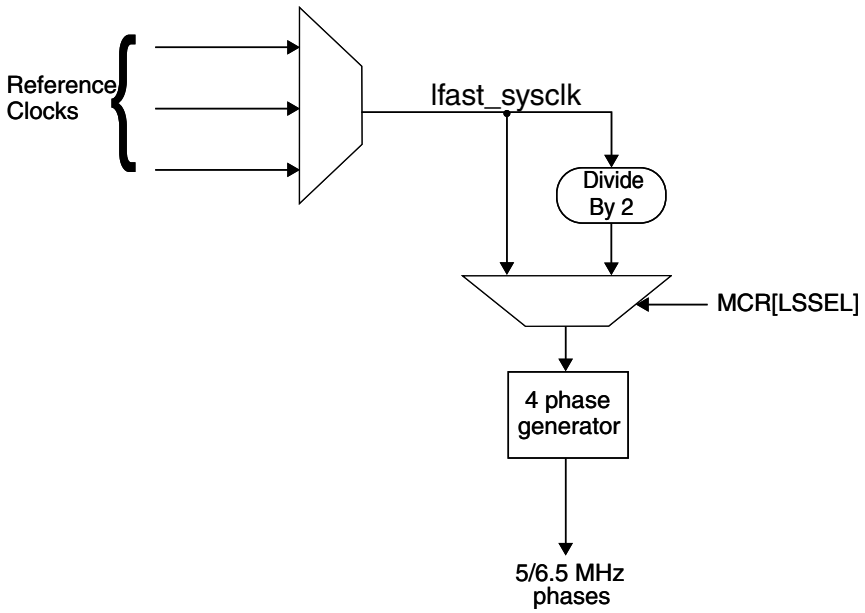


Figure 12-22. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in Figure 12-22.

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 12-23 (selected clock path shown in green).

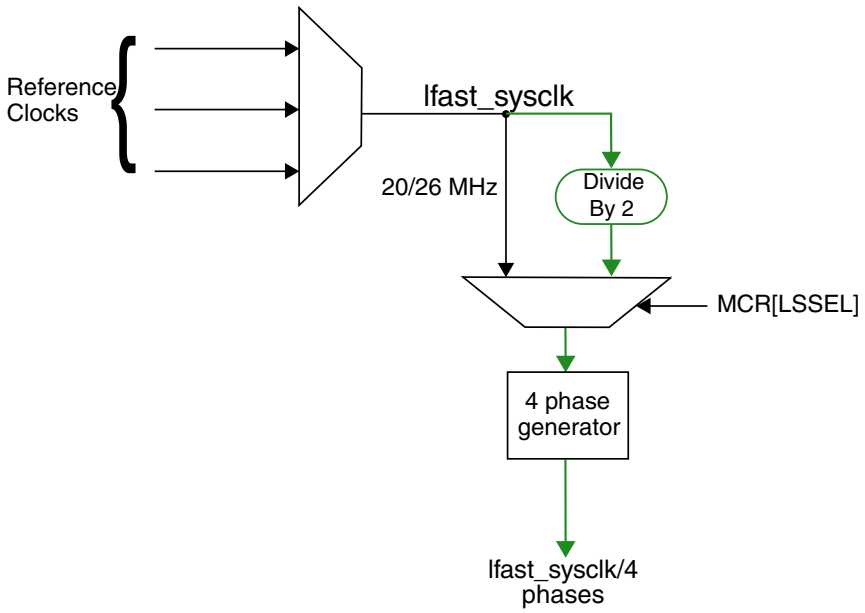


Figure 12-23. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 12-24 (selected clock path selected shown in magenta).

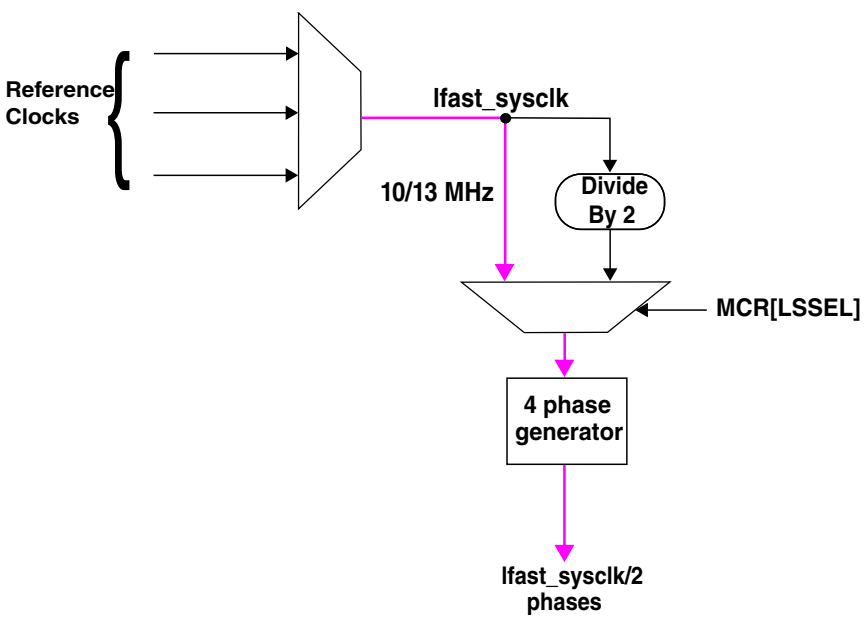


Figure 12-24. External clock muxing of lfast_sysclk of 10 MHz

12.11.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

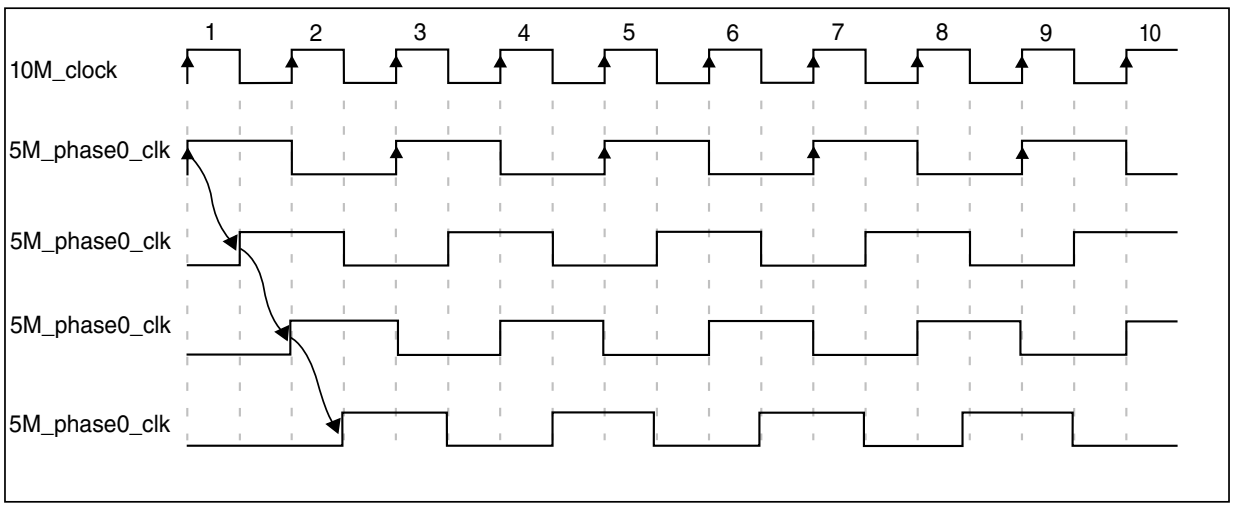


Figure 12-25. Slow speed 4 phases generation

12.11.3 Rx Controller Clocks

12.11.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

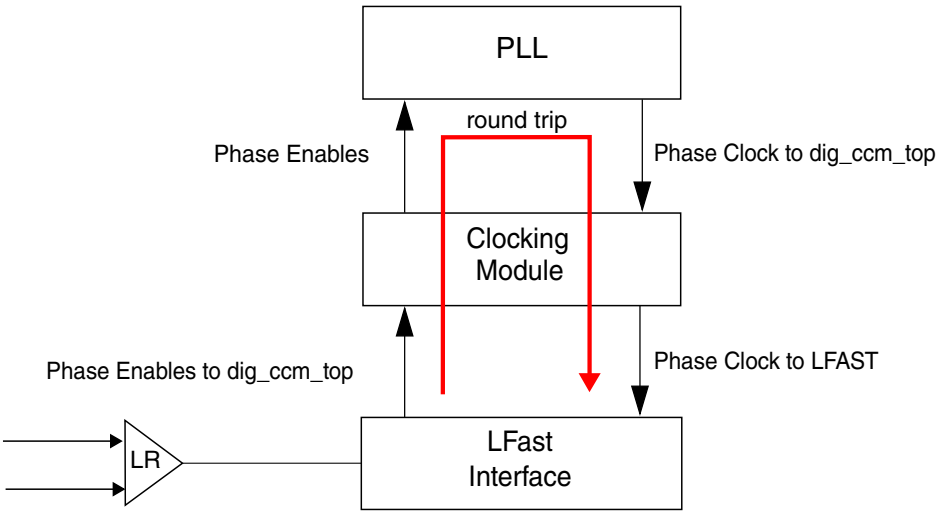


Figure 12-26. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

12.11.5 Clock module requirements for low speed phases

12.11.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 12-12. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

12.11.6 Tx Controller Clocks

12.11.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 12-13. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

12.11.6.2 Transmit Clock Muxing

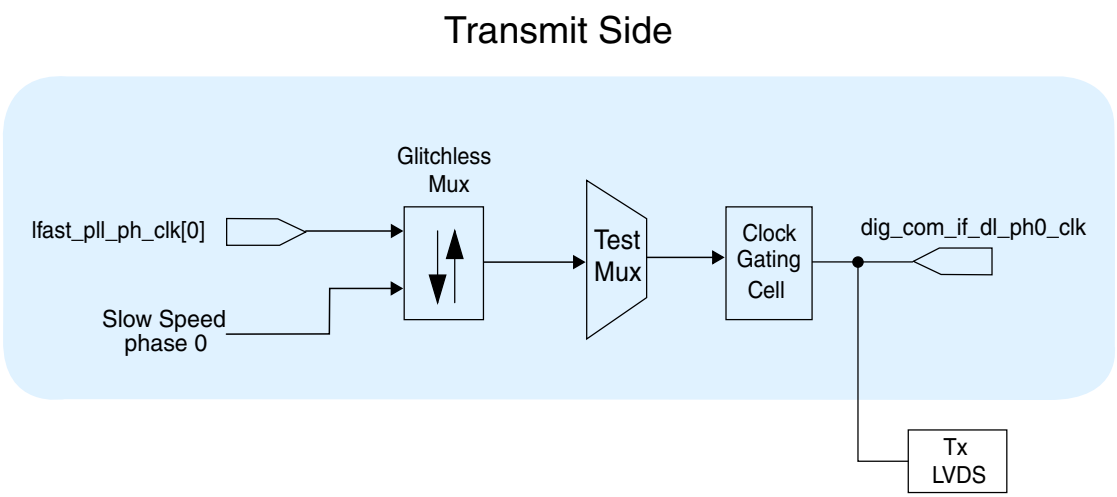


Figure 12-27. Transmit Clocks Muxing

12.11.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

Appendix A
Release Notes for Revision 3

A.1 BD Intro changes

- Minor updates.
- Updated references to IEEE 1149.1 or IEEE 1149.7.
- Updated references of IEEE-ISTO 5001-2011 to IEEE-ISTO 5001-2012.

A.2 Buddy Device SIUL changes

- No substantial content changes

A.3 DCI changes

- In "Software enabled debug and calibration mode" section, minor edit and removed second paragraph of NOTE.
- Removed the following sections: Break control, Cross-triggering control, Synchronous restart control, and EVTO management.
- Updated references to IEEE 1149.1 and IEEE 1149.1.

- In DCI control register (BD_DCI_CR) and Figure 4-2, updated Buddy Device DCI_CR register name to BD_DCI_CR.

- In System watchdog control , revised this section to say "The BD DCI does not include the Software Watchdog Timer (SWT) disable feature".

A.4 JTAGC module changes

- In the "JTAGC block instructions" section, separated General JTAGC instructions and ACCESS_AUX instructions into separate tables. Moved ACCESS_AUX instruction table to the chip configuration debug information.

- In the Features section, removed text "and the ability to share the TAP" form the JCOMP feature bullet.

- In the General JTAG instructions table, updated the Instruction summary cells to emphasize which instructions assert functional reset.

A.5 JTAGM changes

- Updated JTAGM_SR bit 10 reset value to 1.
- Reformatted JTAGM_SR[dci_status] field into separate fields.
- Reformatted JTAGM_SR[lfast_status] field into separate fields.

A.6 NAR changes

- In Error conditions, added Receive Queue Overrun row to NAR error types table.
- In Application information, added description of example flow of NAR configuration for different cases.
- In Trace Memory fill level and partition configuration register (NAR_AHFPAR) and Global (Development tool client) stall control, NAR_AHFPAR[GSE] field description, updated dci_evti[1] to dci_evti[0].

A.7 NAL changes

- Added GSR[TXONLY] bit field.

A.8 DWPU changes

- Removed introductory text at the beginning of register sections.

A.9 Nexus Aurora Phy changes

- Minor non-tecnical edits
- In Introduction, added io_control signal to High speed Nexus interface block diagram.

A.10 RWA changes

- No substantial content changes

A.11 LFAST module changes

• PLL and LVDS Status Register (PLLLSR)
  • Table, "PLLLSR field descriptions"
  • Swapped the field description and bit field value descriptions for LDPDS and LRPDS. Previously, incorrectly defined

• Section, PLL Control Register (PLLCR)
  • Table, "PLLCR field descriptions"
  • Updated IPTMOD field row with new defintions.
  • Updated REFINV field row with new defintions.
  • Updated LPCFG field row with new defintions.
  • Updated FBDIV field row with new defintions.

• Added Section, 'LFAST Tx Interrupt Status Register (LFAST_TISR)' • Added Section, 'LFAST Rx Interrupt Status Register (LFAST_RISR)' • Added Section, 'LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)'
• LVRXOP   • Replaced the description of the LVRXOP field.
• Data receive   • In the first paragraph, changed "MCR[TXEN]" to "MCR[RXEN]".

• LFAST LVDS Control Register (LFAST_LCR)
  • Added LVTXOP at bit 29 (MSB=0).
  • Updated LVTXOP bitfield reset to 1.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number MPC5777MRM
Revision 4, 04/2015